



PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicants: Sung-Eun PARK et al. Docket: 678-1397 (P11316)

Serial No.: 10/811,547 Dated: April 16, 2004

Filed: March 29, 2004

For: **APPARATUS FOR DECODING AN ERROR CORRECTION CODE IN A COMMUNICATION SYSTEM AND METHOD THEREOF**

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

TRANSMITTAL OF PRIORITY DOCUMENT

Sir:

Enclosed is a certified copy of Korean Appln. No. 2003-20255 filed on March 31, 2003, from which priority is claimed under 35 U.S.C. §119.

Respectfully submitted,

Paul J. Farrell
Registration No. 33,494
Attorney for Applicants

DILWORTH & BARRESE, LLP
333 Earle Ovington Boulevard
Uniondale, New York 11553
(516) 228-8484

CERTIFICATE OF MAILING UNDER 37 C.F.R. § 1.8 (a)

I hereby certify that this correspondence is being deposited with the United States Postal Service as first class mail, postpaid in an envelope, addressed to the: Commissioner of Patents, P.O. Box 1450, Alexandria, VA 22313-1450 on April 16, 2004.

Dated: April 16, 2004

Paul J. Farrell



별첨 사본은 아래 출원의 원본과 동일함을 증명함.

This is to certify that the following application annexed hereto
is a true copy from the records of the Korean Intellectual
Property Office.

출원번호 : 10-2003-0020255
Application Number

출원년월일 : 2003년 03월 31일
Date of Application MAR 31, 2003

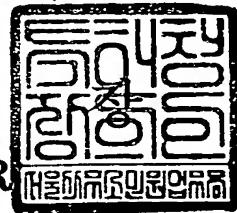
출원인 : 삼성전자주식회사
Applicant(s) SAMSUNG ELECTRONICS CO., LTD.

2004년 03월 05일



특허청

COMMISSIONER



【서지사항】

【서류명】	특허출원서
【권리구분】	특허
【수신처】	특허청장
【참조번호】	0002
【제출일자】	2003.03.31
【국제특허분류】	H04Q
【발명의 명칭】	통신 시스템에서 오류 정정 부호의 복호 장치 및 방법
【발명의 영문명칭】	APPARATUS FOR DECODING ERROR CORRECTION CODE IN COMMUNICATION SYSTEM AND METHOD THEREOF
【출원인】	
【명칭】	삼성전자 주식회사
【출원인코드】	1-1998-104271-3
【대리인】	
【성명】	이건주
【대리인코드】	9-1998-000339-8
【포괄위임등록번호】	2003-001449-1
【발명자】	
【성명의 국문표기】	박성은
【성명의 영문표기】	PARK, Sung Eun
【주민등록번호】	741121-1030911
【우편번호】	442-754
【주소】	경기도 수원시 팔달구 원천동 원천삼성아파트 6동 606호
【국적】	KR
【발명자】	
【성명의 국문표기】	김재열
【성명의 영문표기】	KIM, Jae Yael
【주민등록번호】	700219-1047637
【우편번호】	435-042
【주소】	경기도 군포시 산본2동 산본9단지백두아파트 960동 1401호
【국적】	KR
【발명자】	
【성명의 국문표기】	박성일
【성명의 영문표기】	PARK, Seong Il

【주민등록번호】	680519-1481421		
【우편번호】	463-776		
【주소】	경기도 성남시 분당구 서현동(시범단지) 한양아파트 325동 801호		
【국적】	KR		
【발명자】			
【성명의 국문표기】	김영균		
【성명의 영문표기】	KIM, Young Kyun		
【주민등록번호】	481209-1991001		
【우편번호】	463-010		
【주소】	경기도 성남시 분당구 정자동 상록마을 우성아파트 309동 1405호		
【국적】	KR		
【발명자】			
【성명의 국문표기】	이현우		
【성명의 영문표기】	LEE, Hyeon Woo		
【주민등록번호】	630226-1709811		
【우편번호】	441-390		
【주소】	경기도 수원시 권선구 권선동 벽산아파트 806동 901호		
【국적】	KR		
【취지】	특허법 제42조의 규정에 의하여 위와 같이 출원합니다. 대리인 이건주 (인)		
【수수료】			
【기본출원료】	20	면	29,000 원
【가산출원료】	53	면	53,000 원
【우선권주장료】	0	건	0 원
【심사청구료】	0	항	0 원
【합계】	82,000 원		

【요약서】**【요약】**

본 발명은 k 개의 정보 비트들과 n 개의 출력 비트들을 가지는 (n, k) 블록 부호의 생성 행렬 정보와, 상기 (n, k) 블록 부호를 역 고속 하다마드 변환(IFHT: Inverse Fast Hadamard Transform)하기 위한 IFHT 크기 정보를 가지고 상기 (n, k) 블록 부호를 구성하는 부호화 심볼들 각각을 재배치하기 위한 심볼 위치 정보를 결정하고, 상기 (n, k) 블록 부호를 구성하는 부호화 심볼들 각각을 상기 결정된 심볼 위치 정보에 상응하게 IFHT의 입력으로 재배치하며, 상기 재배치한 심볼들을 입력하여 IFHT를 수행한 후, 상기 IFHT를 수행한 결과들 중에서 최대 상관값을 가지는 (n, k) 블록 부호의 정보 비트를 복호 신호로 출력한다.

【대표도】

도 7

【색인어】

(n, k) 블록 부호, 연판정 복호, 역고속 하다마드 변환, 심볼 위치 정보, 심볼 배치기

【명세서】**【발명의 명칭】**

통신 시스템에서 오류 정정 부호의 복호 장치 및 방법{APPARATUS FOR DECODING ERROR CORRECTION DOE IN COMMUNICATION SYSTEM AND METHOD THEREOF}

【도면의 간단한 설명】

도 1은 일반적인 상관기를 사용하는 연판정 복호 장치 내부 구조를 도시한 도면

도 2는 일반적인 IFHT기를 사용하는 직렬 구조 연판정 복호 장치 내부 구조를 도시한 도면

도 3은 일반적인 IFHT기를 사용하는 병렬 구조 연판정 복호 장치 내부 구조를 도시한 도면

도 4는 일반적인 IFHT 수행 과정을 개략적으로 도시한 도면

도 5는 일반적인 천공된 리드 물러 부호를 IFHT기를 사용하여 복호하는 연판정 복호 장치 내부 구조를 도시한 도면

도 6은 일반적인 반복된 리드 물러 부호를 IFHT를 사용하여 복호하는 연판정 복호 장치 내부 구조를 도시한 도면

도 7은 본 발명의 제1실시예에 따른 IFHT기를 사용한 연판정 복호 장치 내부 구조를 도시한 도면

도 8은 도 7의 제어기(700)의 심볼 위치 정보 결정 과정 및 심볼 배치기(720)의 수신 심볼 재배치 과정을 개략적으로 도시한 도면

도 9는 도 7의 심볼 배치기(720)의 내부 구조를 도시한 도면

도 10은 본 발명의 제2실시예에 따른 IFHT기를 사용한 연판정 복호 장치 내부 구조를 도시한 도면

도 11은 도 10의 제어기(1000)의 심볼 위치 정보 결정 과정 및 심볼 배치기(1020)의 수신 심볼 재배치 과정을 개략적으로 도시한 도면

【발명의 상세한 설명】

【발명의 목적】

【발명이 속하는 기술분야 및 그 분야의 종래기술】

<12> 본 발명은 통신 시스템의 오류 정정 부호 복호 장치 및 방법에 관한 것으로서, 특히 임의의 정보 비트 길이와 블록 길이를 가지는 블록 부호를 복호하는 복호 장치 및 방법에 관한 것이다.

<13> 통상적으로 부호 분할 다중 접속(CDMA: Code Division Multiple Access, 이하 "CDMA"라 칭하기로 한다) 통신 시스템은 전송 채널에서 발생하는 잡음으로 인한 오류를 정정하기 위한 오류 정정(error correction)을 수행한다. 일반적으로 상기 오류 정정을 위한 오류 정정 방식은 송신측에서 정보 비트(information bits)를 오류 정정 기법, 즉 부호화 방식(coding scheme)을 사용하여 부호화한 부호어(codeword)를 수신측으로 송신하고, 상기 수신측이 상기 송신측에서 송신한 부호어를 수신한 후, 상기 송신측에서 적용한 부호화 방식에 상응하는 복호화

방식(decoding scheme)을 사용하여 상기 수신된 부호어를 복호화하여 원래의 정보 비트로 복원하는 방식을 나타낸다. 상기 CDMA 통신 시스템에서 사용되는 대표적인 오류 정정 방식은 블록 부호(block code)를 사용하는 방식과 트렐리스 부호(trellis code) 부호를 사용하는 방식의 2가지 방식들이 존재한다.

- <14> 첫 번째로, 상기 블록 부호를 사용하는 오류 정정 방식을 설명하기로 한다.
- <15> 상기 블록 부호를 사용하는 오류 정정 방식은 일정한 길이의 송신 정보 비트들, 일 예로 k 개의 비트(k bits)에 추가 비트, 일 예로 r 개의 비트(r bits)를 삽입하여 n ($n = k + r$) 비트의 블록 부호로 부호화하여 송신하는 방식으로서, 송신측은 k 비트의 정보 비트를 전송하기 위해서 n 비트의 블록 부호, 즉 (n, k) 블록 부호를 송신한다. 그러면 수신측은 상기 송신측에서 송신한 (n, k) 블록 부호를 수신하고, 상기 수신한 (n, k) 블록 부호를 복호화하여 원래의 k 비트 정보 비트를 추출한다. 또한, 상기 블록 부호를 사용하는 오류 정정 방식의 경우 오류 정정 능력을 증가시키기 위해서는 상기 추가 비트수를 증가시키면 된다. 그리고, 상기 블록 부호의 경우 그 부호어의 크기가 변하게 되면 부호기(encoder) 및 복호기(decoder) 구조가 변경되며, 따라서 동일한 한 시스템에서 서로 다른 길이의 블록 부호를 사용할 경우 상기 서로 다른 길이의 블록 부호를 위한 부호기 및 복호기가 별도로 구비되어야만 한다는 문제점이 있다. 여기서, 상기 블록 부호의 대표적 예로는 BCH 부호, 리드 솔로몬(Reed-Solomon) 부호 등이 있으며, 상기 블록 부호는 Berlekamp-Massey 알고리즘(algorithm), 유클리드 알고리즘 등을 이용하여 경판정(hard decision) 복호된다.
- <16> 두 번째로, 상기 트렐리스 부호를 사용하는 오류 정정 방식을 설명하기로 한다.
- <17> 상기 트렐리스 부호를 사용하는 오류 정정 방식은 송신 정보 비트를 블록으로 세그멘테이션(segmentation)하여 처리하는 방식이 아니라, 쉬프트 레지스터(shift register)에 순서대

로 입력시켜 미리 결정되어 있는 로직(logic)들을 통해 트렐리스 부호로 부호화하여 전송하는 방식이다. 상기 입력되는 송신 정보 비트당 출력 비트수의 비율을 부호화율(coding rate)이라고 하며, 송신측은 상기 부호화율에 상응하게, 일 예로 부호화율이 $1/k$ 일 경우 1비트의 정보 비트를 k 비트의 출력 비트로 부호화하여 송신한다. 그러면 수신측은 상기 송신측에서 송신한 부호화율 $1/k$ 의 트렐리스 부호를 수신하고, 상기 수신한 부호화율 $1/k$ 의 트렐리스 부호를 복호하여 원래의 k 비트 정보 비트를 추출한다. 상기 트렐리스 부호를 사용하는 오류 정정 방식의 경우 오류 정정 능력을 증가시키기 위해서는 부호화율을 감소시켜야 한다. 여기서, 상기 트렐리스 부호의 대표적 예로는 컨벌루셔널(convolutional) 부호와, 터보(turbo) 부호 등이 있으며, 상기 트렐리스 부호는 비터비(Viterbi) 알고리즘 등을 이용하여 연판정(soft decision) 복호된다.

<18> 상기에서 설명한 바와 같이 블록 부호는 일반적으로 복호 과정에서 경판정 복호되는데, 상기 경판정 복호는 수신 신호를 1 혹은 -1로만 판단하여 복호를 수행함으로써 그 복호 성능(decoding performance)이 일반적으로 상기 연판정 복호에 비해서 저하된다. 또한, 상기에서 설명한 바와 같이 트렐리스 부호는 일반적으로 복호 과정에서 연판정 복호되는데, 상기 연판정 복호는 수신 신호를 그 가중치(weight value)에 따라 판단하여 복호를 수행함으로써 상기 경판정 복호에 비해 그 복호 성능이 향상된다. 그러나, 상기 연판정 복호는 수신 신호를 상기 경판정 복호와 같이 단순하게 1 혹은 -1로만 판단하여 복호를 수행하는 것이 아니라 가중치를 고려하여 복호를 수행하기 때문에 그 복호 과정의 연상량이 크게 증가되며, 하드웨어(hardware)적인 복잡도(complexity)도 역시 크게 증가된다. 이런 이유로, 수신되는 블록의 길이, 즉 비트수가 어느 정도 이상 커지면 상기 연판정 복호를 적용하는 것은 난이하다.

<19> 이렇게 연판정 복호 방식이 경판정 복호 방식보다 그 복호 성능이 뛰어나기 때문에 현재 CDMA 통신 시스템은 비교적 그 블록 길이가 짧은 제어 신호(control signal)의 경우 블록 부호를 사용하고, 비교적 그 블록 길이가 긴 정보 신호(information signal)의 경우 트렐리스 부호, 즉 컨벌루셔널 부호 혹은 터보 부호를 사용함으로써 연판정 복호를 구현하고 있다.

<20> 그러면 여기서 도 1을 사용하여 블록 부호를 사용하는 통신 시스템에서 상관기 (correlator)를 사용하는 연판정 복호 장치를 설명하기로 한다.

<21> 상기 도 1은 일반적인 상관기를 사용하는 연판정 복호 장치 내부 구조를 도시한 도면이다.

<22> 상기 도 1을 참조하면, 먼저 수신측으로 수신되는 수신 신호 r 은 상관기(100)로 입력된다. 여기서, 송신측은 임의의 블록 부호를 이진 위상 쉬프트 키잉(BPSK: Binary Phase Shift Keying, 이하 "BPSK"라 칭하기로 한다) 방식을 사용하여 변조한 변조 신호를 송신하였다고 가정하기로 하며, 일 예로 $\{+1, -1\}$ 의 변조 신호를 송신하였다고 가정하기로 한다. 그러면 상기 수신 신호 r 은 상기 송신측에서 송신한 $\{+1, -1\}$ 의 변조 신호가 채널 상황을 겪으면서 잡음 (noise) 성분과 간섭(interference) 성분이 가산된 신호가 되며, 따라서 상기 수신 신호 r 은 $\{+1, -1\}$ 이 아닌 실수값을 가지게 된다. 상기 상관기(100)는 상기 수신 신호 r 을 입력하여 상기 통신 시스템의 송신측에서 송신 가능한 블록 부호의 가능한 모든 부호어들 각각에 대해서 상관한 후, 상기 수신 신호 r 과 상기 모든 부호어들 각각과의 상관값(correlation value)들을 비교 선택기(comparator & selector)(110)로 출력한다. 상기 비교 선택기(110)는 상기 상관기(100)에서 출력한 상기 수신 신호 r 과 상기 모든 부호어들 각각과의 상관값들을 비교하고, 상기 비교 결과 최대 상관값을 가지는 부호어를 선택하여 상기 선택한 부호어를 상기 송신측에서

전송한 부호어로 판단한다. 결과적으로, 상기 비교 선택기(110)에서 출력한 부호어에 상응하는 정보 비트가 원래의 정보 비트로 복원되는 것이다.

<23> 그러면 여기서, 상기 도 1의 연판정 복호 장치에서 (n, k) 블록 부호를 복호화하는 과정을 설명하면 다음과 같다.

<24> 먼저, 송신측에서 (n, k) 블록 부호를 송신하였을 경우, 수신측은 상기 (n, k) 블록 부호에 잡음 성분 및 간섭 성분이 포함된 실수 성분의 수신 신호 r 을 수신하게 되고, 상기 실수 성분의 수신 신호 r 은 상기 상관기(100)로 제공된다. 상기 상관기(100)는 상기 송신측에서 송신 가능한, (n, k) 블록 부호의 모든 부호어들 각각에 대해서 상기 수신 신호 r 과 상관을 수행하고, 그 상관 결과들을 상기 비교 선택기(110)로 출력한다. 여기서, 상기 상관기(100)의 상관 동작에 따른 연산량을 고려하면 다음과 같다.

<25> 먼저, (n, k) 블록 부호의 발생 가능한 모든 부호어들을 고려하면 길이 n 의 부호어가 2^k 개 존재하고, 상기 2^k 개의 길이 n 의 부호어들 각각에 대해서 상관을 수행해야하므로

$n \times 2^k$ 번의 곱셈 과정과, $(n-1) \times 2^k$ 번의 덧셈 과정이 필요로 하게 된다. 일 예로, (n, k) 블록 부호를 $(10, 3)$ 블록 부호라고 가정하면, 상기 $(10, 3)$ 블록 부호의 발생 가능한 모든 부호어들은 8개의 길이 10의 부호어들이며, 상기 8개의 $(10, 3)$ 블록 부호의 부호어를 각각에 대해서 상관을 수행해야 하므로 $10 \times 8 = 80$ 번의 곱셈 과정과, $9 \times 8 = 72$ 번의 덧셈 과정이 필요로 된다. 상기 k 및 n 값이 커질 경우, 특히 상기 k 값이 커질 경우 상기 상관 수행을 위한 곱셈 과정 및 덧셈 과정의 횟수는 기하 급수적으로 증가하게 되며, 결과적으로 연산 과정의 로드로 인해 전체 시스템의 성능이 저하된다는 문제점을 가진다. 그래서, 일반적으로 블록 부호의 경우 연판정 복호를 적용하기 위해서는 그 정보 비트, 즉 k 의 길이에 한정성(일 예로, 14비트 이하)을 가지기 때문에, 연판정 복호의 성능이 경판정 복호 성능 대비 뛰어난 효과를 가짐에도 적용하지 못하는 경우가 발생하게 된다.

<26> 상기 도 1에서는 상관기를 사용하는 연판정 복호 장치 내부 구조를 설명하였으며, 다음으로 도 2를 참조하여 역 고속 하다마드 변환(IFHT: Inverse Fast Hadamard Transform, 이하 "IFHT"라 칭하기로 한다)을 수행하는 역 고속 하다마드 변환기(이하 "IFHT기"라 칭하기로 한다)를 사용하는 연판정 복호 장치를 설명하기로 한다.

<27> 상기 도 2는 일반적인 IFHT기를 사용하는 직렬 구조 연판정 복호 장치 내부 구조를 도시한 도면이다.

<28> 상기 도 2를 설명하기에 앞서, 상기 직렬 구조라 함은 하기에서 설명할 마스크(mask) M_i 를 순차적으로 고려하는 형태를 나타낸다. 이와는 달리 하기에서는 도 3을 참조하여 IFHT기를 사용하는 병렬 구조 연판정 복호 장치를 설명할 것인데, 이 경우 상기 병렬 구조라 함은 상기 마스크 M_i 를 일시적으로 병렬 고려하는 형태를 나타낸다. 상기 도 2를 참조하면, 먼저 수신측

으로 수신되는 수신 신호 r 은 마스크 곱셈기(mask multiplier)(210)로 입력된다. 여기서, 상기 수신측에 상응하는 송신측은 생성 행렬(generator matrix)이 월시(Walsh) 부호의 기저(basis)들을 포함하는 블록 부호를 송신한다. 상기 마스크 곱셈기(210)는 상기 수신 신호 r 과 제어기(controller)(200)에서 출력하는 마스크 M_i 를 곱한 후 IFHT기(220)로 출력한다. 상기 IFHT기(220)는 상기 마스크 곱셈기(210)에서 출력한 신호를 입력하여 IFHT를 수행한 후 그 결과를 비교 선택기(230)로 출력한다. 여기서, 상기 도 2의 연판정 복호 장치는 직렬 구조를 가지므로 상기 제어기(200)는 최초에는 마스크가 적용되지 않는 경우를 가정하여 상기 마스크 M_i 를 출력하지 않고, 이후 상기 제어기(200)는 순차적으로 해당하는 마스크 M_i 를 상기 마스크 곱셈기(210)로 출력한다. 일 예로, 마스크가 M_1 과 M_2 의 2개가 존재할 경우, 상기 제어기는 최초에는 마스크 M_i 를 적용하지 않고, 이후부터 순차적으로 상기 마스크 M_1 과, 마스크 M_2 와, 상기 마스크 M_1 과 마스크 M_2 의 배타적 논리합(XOR)을 상기 마스크 곱셈기(210)로 출력한다.

<29> 그래서, 상기 IFHT기(220)는 상기 마스크 곱셈기(210)에서 출력한 모든 신호들, 즉 마스크가 적용되지 않은 신호, 즉 수신 신호 r 과, 상기 수신 신호 r 과 마스크 M_1 이 곱해진 신호와, 상기 수신 신호 r 과 마스크 M_2 가 곱해진 신호와, 상기 수신 신호 r 과 마스크 M_1 과 마스크 M_2 의 배타적 논리합이 곱해진 신호들 각각에 대해서 순차적으로 IFHT를 수행하고, 그 결과를 상기 비교 선택기(230)로 출력한다. 상기 비교 선택기(230)는 상기 IFHT기(220)에서 출력한 모든 IFHT 결과값들을 비교하여 최대 상관값을 가지는 부호어를 선택하여 상기 선택한 부호어를 상기 송신측에서 전송한 부호어로 판단한다. 결과적으로, 상기 비교 선택기(220)에서 출력한 부호어에 상응하는 정보 비트가 원래의 정보 비트로 복원되는 것이다. 여기서, 상기 제어기(200) 및 마스크 곱셈기(210)의 동작은 하기에서 설명할 것이므로 여기서는 그 상세한 설명을 생략하기로 한다.

<30> 그러면 여기서 (n, k) 리드 뮬러(Reed-Muller) 부호, 일 예로 $(8, 3)$ 리드 뮬러 부호에 대해서 설명하기로 하며, 상기 $(8, 3)$ 리드 뮬러 부호는 하기 표 1과 같다.

<31> 【표 1】

정보 비트	부호어
000	00000000
001	01010101
010	00110011
011	01100110
100	00001111
101	01011010
110	00111100
111	01101001

<32> 상기 표 1에 나타낸 바와 같이, 3비트의 정보 비트가 입력될 때 발생 가능한 $(8, 3)$ 리드 뮬러 부호의 부호어는 2^3 개, 즉 8개이며, 정보 비트가 "000"일 경우에는 "00000000"의 부호어가, 정보 비트가 "001"일 경우에는 "01010101"의 부호어가, 정보 비트가 "010"일 경우에는 "00110011"의 부호어가, 정보 비트가 "011"일 경우에는 "01100110"의 부호어가, 정보 비트가 "100"일 경우에는 "00001111"의 부호어가, 정보 비트가 "101"일 경우에는 "01011010"의 부호어가, 정보 비트가 "110"일 경우에는 "00111100"의 부호어가, 정보 비트가 "111"일 경우에는 "01101001"의 부호어가 생성된다.

<33> 상기 표 1과 같은 $(8, 3)$ 리드 뮬러 부호의 생성 행렬은 하기 수학식 1과 같다.

<34> 【수학식 1】
$$G = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

<35> 상기 수학식 1에서 G 는 생성 행렬을 나타내며, 상기 수학식 1의 생성 행렬에서 행(row)의 수가 입력 정보 비트수 k 와 동일하며, 열(column)의 수가 상기 출력 비트수 n 과 동일하므로

, 상기 생성 행렬에 따라 생성되는 리드 뮤터 부호는 (8,3) 리드 뮤터 부호가 되는 것이다. 상기 생성 행렬의 행들 각각이 기저이며, 따라서 상기 생성 행렬에는 3개의 기저들이 존재한다.

<36> 그러면 여기서, 상기 (8,3) 리드 뮤터 부호를 복호화하는 과정에서 발생하는 연산량을 고려하면 다음과 같다.

<37> 먼저, 상기 (8,3) 리드 뮤터 부호에 대한 모든 부호어들을 고려하면 길이 8의 부호어가 8개 존재하고, 상기 8개의 길이 8의 부호어들 각각에 대해서 IFHT를 수행해야 한다. 상기 (8,3) 리드 뮤터 부호를 가지고 IFHT를 수행하는 과정을 도 4를 참조하여 설명하기로 한다.

<38> 상기 도 4는 일반적인 IFHT 수행 과정을 개략적으로 도시한 도면이다.

<39> 상기 도 4에는 상기의 예와 같이 (8,3) 리드 뮤터 부호를 고려한 경우의 IFHT 수행 과정이 도시되어 있다. 상기 도 4를 참조하면, 먼저 수신 신호 r 은 (8,3) 리드 뮤터 부호의 부호어에 잡음 및 간섭 성분이 삽입된 신호이므로 하기와 같이 표현하기로 한다.

<40> $r = r_1 \ r_2 \ r_3 \ r_4 \ r_5 \ r_6 \ r_7 \ r_8$

<41> 여기서, 상기 r_1 내지 r_8 각각을 수신 심볼이라고 칭하기로 한다. 그리고, 상기 IFHT가 상기 도 1에서 설명한 상관기(100)와 같이 100% 성능의 연판정을 하기 위해서는 상기 (8, 3) 리드 뮤터 부호에서 발생 가능한 모든 부호어들 각각과 상기 수신 신호 r 과의 상관을 고려해야 한다. 결국, 100% 성능의 연판정을 한다는 것은 결국 수신 신호 r 에 대해서 모든 부호어들 각각에 대한 상관을 수행한다는 것을 나타내는 것이다. 실제 상기 (8, 3) 리드 뮤터 부호는 상기 표 1에 나타낸 바와 같지만, 이는 디지털 데이터(digital data)로서 표현한 것이며, 실제 에어(air)상에서 상기 디지털 데이터들은 일 예로 BPSK 방식으로 변조되기 때문에 상기 디지털

데이터 "0"은 "+1"로, 상기 디지털 데이터 "1"은 "-1"로 대응되어 전송된다. 따라서, 상기 표 1에서 나타낸 (8, 3) 리드 물러 부호를 BPSK 변조된 성분들로 대응시키면 하기 표 2와 같다.

<42> 【표 2】

정보 비트	BPSK 방식으로 변조된 부호어 성분
000	++++++
001	+---++-
010	+-+-+--
011	+-+-++-
100	++++--
101	+---++-
110	+-+-++
111	+-+-+-++

<43> 상기 100% 연관성을 위해서는 상기 수신 신호 r , 즉 $r_1 r_2 r_3 r_4 r_5 r_6 r_7 r_8$ 과 상기 표 2에 나타낸 바와 같은 BPSK 방식으로 변조된 모든 부호어들 각각에 대한 상관을 수행해야한다. 상기 IFHT를 수행하면 상기 수신 신호 r 과 상기 BPSK 방식으로 변조된 모든 부호어들 각각에 대한 상관이 가능한데, 이는 상기 도 4에 나타낸 바와 같이 나비 로직(butterfly logic) 구조의 IFHT가 수행되기 때문이다.

<44> 즉, 수신 신호 $r_1 r_2 r_3 r_4 r_5 r_6 r_7 r_8$ 에 대해서 2의 역승 단위로 스테이지(stage)들을 입력 정보 비트수인 k 번, 즉 3번 수행하면 상기 BPSK 방식으로 변조된 모든 부호어들 각각에 대한 상관이 가능하다. 여기서, 상기 스테이지를 각각은 또 다시 2의 역승 단위로 상기 수신 신호 r 의 성분들 각각에 대해서 + 연산

과, - 연산을 수행한다. 이를 자세히 설명하면, 첫 번째로, 제1스테이지에서는 상기 수신 신호 r 의 성분들 각각에 대해서 $2^0(=1)$ 단위로 + 연산과 - 연산을 수행한다. 즉, r_1 과 r_2 간에 대해서 각각 + 연산과 -연산을 수행하고, r_3 과 r_4 간에 대해서 각각 + 연산과 -연산을 수행하고, r_5 와 r_6 간에 대해서 각각 + 연산과 -연산을 수행하고, r_7 과 r_8 간에 대해서 각각 + 연산과 -연산을 수행한다. 두 번째로, 제2스테이지에서는 상기 제1스테이지의 결과 성분들, 즉 r_1+r_2 , r_1-r_2 , r_3+r_4 , r_3-r_4 , r_5+r_6 , r_5-r_6 , r_7+r_8 , r_7-r_8 각각에 대해서 $2^1(=2)$ 단위로 + 연산과 -연산을 수행한다. 즉, r_1+r_2 와 r_3+r_4 간에 대해서 각각 + 연산과 -연산을 수행하고, r_1-r_2 와 r_3-r_4 간에 대해서 각각 + 연산과 -연산을 수행하고, r_5+r_6 와 r_7+r_8 간에 대해서 각각 + 연산과 -연산을 수행하고, r_5-r_6 과 r_7-r_8 간에 대해서 각각 + 연산과 -연산을 수행한다. 세 번째로, 제3스테이지에서는 상기 제2스테이지의 결과 성분들, 즉 $(r_1+r_2)+(r_3+r_4)$ 와, $(r_1-r_2)+(r_3-r_4)$ 와, $(r_1+r_2)-(r_3+r_4)$ 와, $(r_1-r_2)-(r_3-r_4)$ 와, $(r_5+r_6)+(r_7+r_8)$ 와, $(r_5-r_6)+(r_7-r_8)$ 와, $(r_5+r_6)-(r_7+r_8)$ 와, $(r_5-r_6)-(r_7-r_8)$ 각각에 대해서 $2^2(=4)$ 단위로 + 연산과 -연산을 수행한다. 즉, $(r_1+r_2)+(r_3+r_4)$ 와 $(r_5+r_6)+(r_7+r_8)$ 간에 대해서 각각 + 연산과 -연산을 수행하고, $(r_1-r_2)+(r_3-r_4)$ 와 $(r_5-r_6)+(r_7-r_8)$ 간에 대해서 각각 + 연산과 -연산을 수행하고, $(r_1+r_2)-(r_3+r_4)$ 와 $(r_5+r_6)-(r_7+r_8)$ 간에 대해서 각각 + 연산과 -연산을 수행하고, $(r_1-r_2)-(r_3-r_4)$ 와 $(r_5-r_6)-(r_7-r_8)$ 간에 대해서 각각 + 연산과 -연산을 수행한다. 이렇게 제3스테이지까지 수행한 결과를 보면, 수신 신호 $r_1 r_2 r_3 r_4 r_5 r_6 r_7 r_8$ 에 대해서 상기 표 2에서 설명한 BPSK 방식으로 변조된 모든 부호어를 각각에 대한 상관이 수행된 것을 알 수 있다.

<45> 즉, 상기 수신 신호 $r_1 r_2 r_3 r_4 r_5 r_6 r_7 r_8$ 에 대해서 상기 표 2의 BPSK 방식으로 변조된 첫 번째 부호어, 즉 +++++++에 대한 상관 결과가 $((r_1+r_2)+(r_3+r_4))+((r_5+r_6)+(r_7+r_8))$ 이며, 두 번째 부호어, 즉 +-+-+-+에 대한 상관 결과가 $((r_1-r_2)+(r_3-r_4))+((r_5-r_6)+(r_7-r_8))$ 이다.

며, 세 번째 부호어, 즉 $++--++-$ 에 대한 상관 결과가 $((r_1+r_2)-(r_3+r_4)) + ((r_5+r_6)-(r_7+r_8))$ 이며, 네 번째 부호어, 즉 $--++-$ 에 대한 상관 결과가 $((r_1-r_2)-(r_3-r_4)) + ((r_5-r_6)-(r_7-r_8))$ 이며, 다섯 번째 부호어, 즉 $++++-$ 에 대한 상관 결과가 $((r_1+r_2)+(r_3+r_4)) - ((r_5+r_6)+(r_7+r_8))$ 이며, 여섯 번째 부호어, 즉 $+-+-+-$ 에 대한 상관 결과가 $((r_1-r_2)+(r_3-r_4)) - ((r_5-r_6)+(r_7-r_8))$ 이며, 일곱 번째 부호어, 즉 $-----++$ 에 대한 상관 결과가 $((r_1+r_2)-(r_3+r_4)) - ((r_5+r_6)-(r_7+r_8))$ 이며, 여덟 번째 부호어, 즉 $+--+---$ 에 대한 상관 결과가 $((r_1-r_2)-(r_3-r_4)) - ((r_5-r_6)-(r_7-r_8))$ 이다. 결과적으로, 상기 수신 신호 $r_1 \ r_2 \ r_3 \ r_4 \ r_5 \ r_6 \ r_7 \ r_8$ 에 대해서 상기 표 2의 BPSK 방식으로 변조된 모든 $(8,3)$ 리드 뮤터 부호의 부호어들에 대한 상관이 100% 수행된 것이다.

<46> 그러면 여기서, 상기 $(8, 3)$ 리드 뮤터 부호의 IFHT 수행에 따른 연산량을 살펴보면, $\log_2 8 (=3)$ 번의 스테이지들을 통해 $8 \log_2 8 (=24)$ 번의 덧셈 (+ 연산, - 연산) 과정, 즉 제1스테이지에서 8번의 덧셈 과정과, 제2스테이지에서 8번의 덧셈 과정과, 제3스테이지에서 8번의 덧셈 과정의 총 24번의 덧셈 과정이 필요하게 된다. 한편, 상기 $(8, 3)$ 리드 뮤터 부호를 상관기를 통해 상관할 경우 연산량을 살펴보면, $8 \times 2^3 (=64)$ 번의 곱셈 과정과, $(8-1) \times 2^3 (=56)$ 번의 덧셈 과정이 필요하게 된다.

<47> 결국, (n, k) 블록 부호에 대한 모든 부호어들을 고려하면 길이 n 의 부호어가 2^k 개 존재하고, 상기 2^k 개의 길이 n 의 부호어들 각각에 대해서 상관을 수행해야 하므로 상관기를 통해 상관을 수행할 경우에는 $n \times 2^k$ 번의 곱셈 과정과, $(n-1) \times 2^k$ 번의 덧셈 과정이 필요하게 된다. 이와는 달리 (n, k) 블록 부호에 대한 모든 부호어들을 IFHT를 수행할 경우에는 $n \log_2 n$ 의 덧셈 과정이 필요하게 된다. 결과적으로, 모든 블록 부호들에 대해서 IFHT를 통해 연산정 복호를 할

경우에는 100% 상관을 고려한 연판정 복호가 가능하게 되면서도, 연산량을 최소화하여 연판정 복호 성능이 극대화된다.

<48> 다음으로, 상기 도 2의 제어기(200) 및 마스크 곱셈기(210)에 대해서 설명하기로 한다.

<49> 먼저, 리드 뮤터 부호에 마스크가 적용된다는 것은 생성 행렬의 기저에 마스크로 사용될 기저가 추가됨을 나타낸다. 즉, 상기에서 설명한 바와 같이 생성 행렬은 입력 정보 비트수와 동일한 개수의 기저들을 가지는데 마스크가 적용될 경우에는 상기 생성 행렬은 상기 입력 정보 비트수와 동일한 개수의 기저들 뿐만 아니라 상기 마스크로 사용될 기저들을 가진다.

<50> 일 예로, 상기 표 1과 같은 (8, 3) 리드 뮤터 부호에 all-one 마스크가 적용되는 경우를 고려하면 생성 행렬은 하기 수학식 2와 같다.

<51>

$$G = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

【수학식 2】

<52> 상기 수학식 2에서 G는 생성 행렬을 나타내며, 상기 수학식 2의 생성 행렬의 제4행의 all-one 기저가 상기 (8, 3) 리드 뮤터 부호의 마스크 기저이다.

<53> 상기 제어기(200)는 상기 수신 신호 r 이 최초에 수신되면, 먼저 마스크가 적용되지 않았다고 가정하고, 상기 마스크 M_i 를 출력하지 않도록 제어하여 상기 IFHT기(220)가 상기 수신 신호 r 을 그대로 IFHT를 수행하도록 제어한다. 여기서, 상기 제어기(200)는 상기에서 설명한 바와 같이 최초에는 마스크를 고려하지 않고, 이후 다시 마스크 M_i 를 고려할 수도 있으며, 이와는 달리 상기 마스크가 적용되지 않은 경우 all-one 마스크를 적용하였다고 가정하여 상기 마스크 곱셈기(210)으로 모든 엘리먼트(element)들이 1로 이루어진 마스크 M_i 를 출력할 수도 있다. 상기 모든 엘리먼트들이 1로 이루어진 마스크는 실질적으로 수신 신호와 곱해져도 변화가

발생되지 않아 마스크가 적용되지 않은 경우와 동일하게 동작한다. 이렇게 모든 엘리먼트들이 1로 이루어진 마스크 M_i 를 사용함으로써 실제 마스크가 적용된 경우와 마스크가 적용되지 않은 경우의 하드웨어 구조를 동일하게 유지할 수 있게 된다. 그러나, 실질적으로 상기 수신 신호 r 에는 마스크가 적용되었으며 따라서 상기 제어기(200)는 상기 마스크 곱셈기(210)로 상기 마스크 기저에 해당하는 마스크 M_i 를 출력한다, 여기서, 상기 생성 행렬의 마스크 기저는 all-one 기저이므로 상기 마스크 M_i 는 모든 엘리먼트들이 -1로 이루어진다. 상기 마스크 곱셈기(210)는 상기 수신 신호 r 과 상기 마스크 M_i 를 곱한 후 상기 IFHT기(220)로 출력한다. 상기 도 2의 연관정 복호 장치는 직렬 구조를 가지기 때문에 상기 마스크 벡터 M_i 를 고려하는 경우와 고려하지 않는 경우를 순차적으로 고려하였으며, 상기 도 3을 참조하여 병렬 구조의 연관정 복호 장치를 설명하기로 한다.

<54> 상기 도 3은 일반적인 IFHT기를 사용하는 병렬 구조 연관정 복호 장치 내부 구조를 도시한 도면이다.

<55> 상기 도 3을 참조하면, 먼저 수신측으로 수신되는 수신 신호 r 은 IFHT기(311)와, 다수의 마스크 곱셈기들(321, 331, 341)로 입력된다. 상기 송신측에서 마스크로 적용한 기저들의 개수에 따라 수신측에서 구비해야 할 상기 마스크 곱셈기들의 개수가 결정되며, 상기 도 3에서는 상기 마스크로 적용한 기저들의 개수가 2개라고 가정하기로 한다. 그러므로, 상기 수신측에는 제1마스크 기저 m_1 에 대응하는 제1마스크 M_1 을 곱해주는 마스크 곱셈기(321)와, 제2마스크 기저 m_2 에 대응하는 제2마스크 M_2 을 곱해주는 마스크 곱셈기(331)와, 상기 제1마스크 기저 m_1 과 제2마스크 기저 m_2 의 배타적 논리합에 대응하는 마스크(이하 " $M_1 \oplus M_2$ "라 칭하기로 한다)를 곱해주는 마스크 곱셈기(341)이 구비된다.

<56> 상기 IFHT기(311)는 상기 수신 신호 r 을 입력하여 IFHT를 수행한 후 비교 선택기(350)로 출력한다. 또한, 상기 마스크 곱셈기들(321, 331, 341) 각각은 상기 수신 신호 r 을 입력하여 해당하는 마스크, 즉 제1마스크 M_1 과, 제2마스크 M_2 와, 제1마스크 기저 m_1 과 제2마스크 기저 m_2 의 배타적 논리합에 대응하는 마스크 $M_1 \oplus M_2$ 각각을 곱한 후 IFHT기(323) 내지 IFHT기(343) 각각으로 출력한다. 상기 IFHT기들(323, 333, 343) 각각은 상기 마스크 곱셈기들(321, 331, 341) 각각에서 출력한 신호들을 입력하여 IFHT를 수행한 후 상기 비교 선택기(350)로 출력한다. 상기 비교 선택기(350)는 상기 IFHT기들(311, 323, 333, 343) 각각에서 출력하는 IFHT 수행 결과를 비교하여 최대 상관값을 가지는 부호어를 선택하여 상기 선택한 부호어를 상기 송신측에서 전송한 부호어로 판단한다. 결과적으로, 상기 비교 선택기(330)에서 출력한 부호어에 상응하는 정보 비트가 원래의 정보 비트로 복원되는 것이다.

<57> 상기에서 설명한 바와 같이 (n, k) 블록 부호에 대한 모든 부호어들을 고려하면 길이 n 의 부호어가 2^k 개 존재하고, 상기 2^k 개의 길이 n 의 부호어들 각각에 대해서 상관을 수행해야 하므로 상관기를 통해 상관을 수행할 경우에는 $n \times 2^k$ 번의 곱셈 과정과, $(n-1) \times 2^k$ 번의 덧셈 과정이 필요하게 되는 반면에, (n, k) 블록 부호에 대한 모든 부호어들을 IFHT기를 사용하여 상관을 수행할 경우에는 $n \log_2 n$ 의 덧셈 과정만이 필요하게 되어 연산량이 최소화되므로 연관정 복호 성능이 극대화된다. 그러나, 상기 도 1에서 설명한 상관기를 이용한 연관정 복호 장치는 임의의 블록 부호들에 대해서 모두 연관정 복호화를 수행하는 것이 가능하지만, 상기 도 2에서 설명한 IFHT를 사용하는 연관정 복호 장치는 생성 행렬이 월시 부호의 기저를 포함하는 블록 부호들에 대해서만 연관정 복호화를 수행하는 것이 가능하다는 단점을 가진다.

【발명이 이루고자 하는 기술적 과제】

<58> 따라서, 본 발명의 목적은 통신 시스템에서 오류 정정 부호를 복호하는 장치 및 방법을 제공함에 있다.

<59> 본 발명의 다른 목적은 통신 시스템에서 최소 연산량을 가지는 오류 정정 부호 복호 장치 및 방법을 제공함에 있다.

<60> 본 발명의 또 다른 목적은 임의의 정보 비트 길이와 블록 길이를 가지는 블록 부호들에 대해 최소 연산량을 가지는 연관정 복호를 수행하는 장치 및 방법을 제공함에 있다.

<61> 본 발명의 또 다른 목적은 통신 시스템에서 임의의 블록 부호들에 대해 IFHT를 사용하여 연관정 복호를 수행하는 장치 및 방법을 제공함에 있다.

<62> 상기한 목적들을 달성하기 위한 본 발명의 제1장치는; k개의 정보 비트들과 n개의 출력 비트들을 가지는 (n, k) 블록 부호를 복호하는 장치에 있어서, 상기 (n, k) 블록 부호의 생성 행렬 정보와, 상기 (n, k) 블록 부호를 역 고속 하다마드 변환(IFHT: Inverse Fast Hadamard Transform)하기 위한 IFHT 크기 정보를 가지고 상기 (n, k) 블록 부호를 구성하는 부호화 심볼들 각각을 재배치하기 위한 심볼 위치 정보를 결정하는 제어기와, 상기 (n, k) 블록 부호를 구성하는 부호화 심볼들 각각을 상기 제어기가 결정한 심볼 위치 정보에 상응하게 IFHT기의 입력으로 재배치하는 심볼 배치기와, 상기 심볼 배치기에서 재배치한 심볼들을 입력하여 IFHT를 수행하는 IFHT기와, 상기 IFHT를 수행한 결과들중에서 최대 상관값을 가지는 (n, k) 블록 부호의 정보 비트를 복호 신호로 출력하는 비교 선택기를 포함함을 특징으로 한다.

<63> 상기한 목적들을 달성하기 위한 본 발명의 제2장치는; k개의 정보 비트들과 n개의 출력 비트들을 가지는 (n, k) 블록 부호를 복호하는 장치에 있어서, 상기 (n, k) 블록 부호의 생성

행렬 정보를 가지고 상기 (n, k) 블록 부호를 역 고속 하다마드 변환(IFHT: Inverse Fast Hadamard Transform)하기 위한 IFHT 크기 정보와, 상기 (n, k) 블록 부호를 구성하는 부호화 심볼들 각각을 재배치하기 위한 심볼 위치 정보를 결정하는 체어기와, 상기 (n, k) 블록 부호를 구성하는 부호화 심볼들 각각을 상기 제어기가 결정한 심볼 위치 정보에 상응하게 IFHT기의 입력으로 재배치하는 심볼 배치기와, 상기 심볼 배치기에서 재배치한 심볼들을 입력하여 IFHT를 수행하는 IFHT기와, 상기 IFHT를 수행한 결과들중에서 최대 상관값을 가지는 (n, k) 블록 부호의 정보 비트를 복호 신호로 출력하는 비교 선택기를 포함함을 특징으로 한다.

<64> 상기한 목적들을 달성하기 위한 본 발명의 제1방법은; k 개의 정보 비트들과 n 개의 출력 비트들을 가지는 (n, k) 블록 부호를 복호하는 방법에 있어서, 상기 (n, k) 블록 부호의 생성 행렬 정보와, 상기 (n, k) 블록 부호를 역 고속 하다마드 변환(IFHT: Inverse Fast Hadamard Transform)하기 위한 IFHT 크기 정보를 가지고 상기 (n, k) 블록 부호를 구성하는 부호화 심볼들 각각을 재배치하기 위한 심볼 위치 정보를 결정하는 과정과, 상기 (n, k) 블록 부호를 구성하는 부호화 심볼들 각각을 상기 결정된 심볼 위치 정보에 상응하게 IFHT의 입력으로 재배치하는 과정과, 상기 재배치한 심볼들을 입력하여 IFHT를 수행하는 과정과, 상기 IFHT를 수행한 결과들중에서 최대 상관값을 가지는 (n, k) 블록 부호의 정보 비트를 복호 신호로 출력하는 과정을 포함함을 특징으로 한다.

<65> 상기한 목적들을 달성하기 위한 본 발명의 제2방법은; k 개의 정보 비트들과 n 개의 출력 비트들을 가지는 (n, k) 블록 부호를 복호하는 방법에 있어서, 상기 (n, k) 블록 부호의 생성 행렬 정보를 가지고 상기 (n, k) 블록 부호를 역 고속 하다마드 변환(IFHT: Inverse Fast Hadamard Transform)하기 위한 IFHT 크기 정보와, 상기 (n, k) 블록 부호를 구성하는 부호화 심볼들 각각을 재배치하기 위한 심볼 위치 정보를 결정하는 과정과, 상기 (n, k) 블록 부호를

구성하는 부호화 심볼들 각각을 상기 결정된 심볼 위치 정보에 상응하게 IFHT의 입력으로 재배치하는 과정과, 상기 재배치한 심볼들을 입력하여 IFHT를 수행하는 과정과, 상기 IFHT를 수행한 결과들중에서 최대 상관값을 가지는 (n, k) 블록 부호의 정보 비트를 복호 신호로 출력하는 과정을 포함함을 특징으로 한다.

【발명의 구성 및 작용】

<66> 이하, 본 발명에 따른 바람직한 실시예를 첨부한 도면을 참조하여 상세히 설명한다. 하기의 설명에서는 본 발명에 따른 동작을 이해하는데 필요한 부분만이 설명되며 그 이외 부분의 설명은 본 발명의 요지를 훌트리지 않도록 생략될 것이라는 것을 유의하여야 한다.

<67> 먼저, 상기 종래 기술 부분에서 설명한 역 고속 하다마드 변환(IFHT: Inverse Fast Hadamard Transform, 이하 "IFHT"라 칭하기로 한다)에 대해서 다시 한번 설명하기로 한다.

<68> 상기 IFHT를 수행하는 역 고속 하다마드 변환기(이하 "IFHT기"라 칭하기로 한다)를 사용하는 연판정 복호(soft decision decoding)는 상관기(correlator)를 사용하는 연판정 복호와 동일한 연판정 성능을 가지면서도, 상기 상관기의 상관 수행에 따른 연산량(일 예로, (n, k) 블록 부호(block code)일 경우 $n \times 2^k$ 번의 곱셈 과정 및 $(n-1) \times 2^k$ 번의 덧셈 과정이 필요함)보다 훨씬 적은 연산량((n, k) 블록 부호일 경우 $n \log_2 n$ 의 덧셈 과정이 필요함)을 가져 연산 과정으로 인한 로드(load)를 최소화시킨다. 그러나, 상기 IFHT기를 사용하는 연판정 복호는 생성 행렬(generator matrix)이 월시 부호(walsh code)의 기저(basis)를 포함하는 블록 부호들에 대해서만 적용 가능하였었기 때문에, 상기 IFHT기를 사용하는 연판정 복호가 상관기를 사용하

는 연판정 복호에 비해 뛰어난 성능을 가지면서도 사용되지 못하는 경우가 빈번하게 발생하였다.

<69> 그래서, 본 발명은 상기 월시 부호의 기저를 포함하지 않는 블록 부호들에 대해서도 상기 IFHT기를 사용하여 연판정 복호함으로써 복호 성능을 최대화시키는 방안을 제시한다.

<70> 먼저, 상기 IFHT의 특성을 살펴보기로 한다.

<71> 상기 IFHT의 특성을 살펴보기에 앞서, 상기 종래 기술에서 설명한 바와 같이 월시 부호의 기저를 포함하는 블록 부호, 즉 리드 뮬러(Reed-Muller) 부호를 다시 한번 설명하기로 한다. (n, k) 리드 뮬러 부호, 일 예로 $(8,3)$ 리드 뮬러 부호는 상기 종래 기술 부분에서 설명한 표 1과 같다. 여기서, 상기 k 는 입력 정보 비트의 길이를 나타내며, 상기 $n = 2^k$ 은 출력되는 블록 길이를 나타낸다. 즉, 상기 표 1에 나타낸 바와 같이, 3비트의 정보 비트가 입력될 때 발생 가능한 $(8,3)$ 리드 뮬러 부호의 부호어(codeword)는 2³개, 즉 8개이며, 정보 비트가 "000"일 경우에는 "00000000"의 부호어가, 정보 비트가 "001"일 경우에는 "01010101"의 부호어가, 정보 비트가 "010"일 경우에는 "00110011"의 부호어가, 정보 비트가 "011"일 경우에는 "01100110"의 부호어가, 정보 비트가 "100"일 경우에는 "00001111"의 부호어가, 정보 비트가 "101"일 경우에는 "01011010"의 부호어가, 정보 비트가 "110"일 경우에는 "00111100"의 부호어가, 정보 비트가 "111"일 경우에는 "01101001"의 부호어가 생성된다. 그리고, 상기 $(8,3)$ 리드 뮬러 부호의 부호어들 각각은 실제 변조 방식, 일 예로 이진 위상 쉬프트 키잉(BPSK: Binary Phase Shift Keying, 이하 "BPSK"라 칭하기로 한다) 방식으로 변조할 경우 디지털 데이터(digital data) "0"은 "+1"로, 상기 디지털 데이터 "1"은 "-1"로 대응되어 실제 에어(air)상으로 전송된다.

<72> 상기 (8, 3) 리드밀러 부호에서 발생 가능한 모든 부호어들을 상기 BPSK 방식으로 변조할 경우, 상기 "00000000"의 부호어는 "+++++++"로, 상기 "01010101"의 부호어는 "+---+++"로, 상기 "00110011"의 부호어는 "----++"로, 상기 "01100110"의 부호어는 "++--+++"로, 상기 "00001111"의 부호어는 "++++---"로, 상기 "01011010"의 부호어는 "++-+++"로, 상기 "00111100"의 부호어는 "+++++++"로, 상기 "01101001"의 부호어는 "++-+++"로 변조된다. 상기 (8, 3) 리드밀러 부호에서 발생 가능한 모든 부호어들 각각을 상기 BPSK 방식으로 변조하였을 때의 변조 성분들을 나타내면 하기 표 3과 같다.

<73> 【표 3】

+	+	+	+	+	+	+	+
+	-	+	-	+	-	+	-
+	+	-	-	+	+	-	-
+	-	-	+	+	-	-	+
+	+	+	+	-	-	-	-
+	-	+	-	+	-	+	-
+	+	-	-	-	-	+	+
+	-	-	-	+	-	+	-

<74> 상기 표 3에서, 제1행(row)이 "00000000" 부호어의 BPSK 변조 성분에 해당하며, 제2행이 "01010101" 부호어의 BPSK 변조 성분에 해당하며, 제3행이 "00110011" 부호어의 BPSK 변조 성분에 해당하며, 제4행이 "01100110" 부호어의 BPSK 변조 성분에 해당하며, 제5행이 "00001111" 부호어의 BPSK 변조 성분에 해당하며, 제6행이 "01011010" 부호어의 BPSK 변조 성분에 해당하며, 제7행이 "00111100" 부호어의 BPSK 변조 성분에 해당하며, 제8행이 "01101001" 부호어의 BPSK 변조 성분에 해당한다.

<75> 그러면, 여기서 천공(puncturing)된 리드 뮤크 부호, 즉 $(n-t, k)$, 일 예로 $(8, 3)$ 리드 뮤크 부호가 2비트 천공된 $(6, 3)$ 리드 뮤크 부호를 가지고 IFHT를 수행하는 과정을 도 5를 참조하여 설명하기로 한다. 여기서, t 는 천공되는 비트수를 나타낸다.

<76> 상기 도 5는 일반적인 천공된 리드 뮤크 부호를 IFHT기를 사용하여 복호하는 연판정 복호 장치 내부 구조를 도시한 도면이다. 상기 도 5를 설명하기에 앞서, 먼저, 상기 $(6, 3)$ 리드 뮤크 부호는 상기 표 1에서 설명한 $(8, 3)$ 리드 뮤크 부호의 부호어들 각각에서 선행하는 2비트(2bits)씩을 천공한 것이며, 이를 나타내면 하기 표 4와 같다.

<77> 【표 4】

정보 비트	부호어
000	000000
001	010101
010	110011
011	100110
100	001111
101	011010
110	111100
111	101001

<78> 그리고, 상기 $(6, 3)$ 리드 뮤크 부호에서 발생 가능한 모든 부호어들 각각을 상기 BPSK 방식으로 변조하였을 때의 변조 성분들을 나타내면 하기 표 5와 같다.

<79> 【표 5】

+	+	+	+	+	+
+	-	+	-	+	-
-	-	+	+	-	-
-	+	+	-	-	+
+	+	-	-	-	-
+	-	+	-	+	-
-	-	-	-	+	+
-	+	-	+	+	-

<80> 상기 도 5를 참조하면, 먼저 수신 신호 r 은 (6,3) 리드 뮤러 부호에 잡음(noise) 성분 및 간섭(interference) 성분이 삽입된 신호이므로 하기와 같이 나타낸다.

<81> $r = r_1 \ r_2 \ r_3 \ r_4 \ r_5 \ r_6$

<82> 상기 수신 신호 r 은 0 삽입기(0 insertor)(511)로 전달되고, 상기 0 삽입기(511)는 상기 수신 신호 r 을 입력하여 미리 설정된 위치에서 0을 삽입하여 IFHT기(513)로 출력한다. 여기서, 상기 0 삽입기(511)는 송신측에서 (8, 3) 리드 뮤러 부호에서 비트를 천공한 위치에 0을 삽입하는 것이며, 상기 천공 위치 관련 정보는 송신측과 수신측 상호가 인지하고 있다. 상기 IFHT 기(513)는 상기 0 삽입기(511)에서 출력한 신호를 입력하여 IFHT를 수행한 후 그 결과를 비교 선택기(comparator & selector)(515)로 출력한다. 상기 비교 선택기(515)는 상기 IFHT기(513)에서 출력한 모든 IFHT 결과값들을 비교하여 최대 상관값을 가지는 부호어를 선택하여 상기 선택한 부호어를 상기 송신측에서 송신한 부호어로 판단한다. 결과적으로, 상기 비교 선택기(515)에서 출력한 부호어에 상응하는 정보 비트가 원래의 정보 비트로 복원되는 것이다. 상기 IFHT 수행 과정은 상기 종래 기술 부분의 도 4에서 설명하였으므로 여기서는 그 상세한 설명을 생략하기로 한다. 결국, 상기 천공된 $(n-t, k)$ 리드 뮤러 부호에 대해서 IFHT를 사용하여 연판정 복호할 때는 상기 (n, k) 리드 뮤러 부호에서 천공된 위치의 비트들을 모두 0으로 삽입하여 (n, k) 리드 뮤러 부호화하여 IFHT를 사용하여 연판정 복호하는 것과 동일하게 된다. 또한, 연판정 복호를 위한 연산량 역시 상기 (n, k) 리드 뮤러 부호의 연산량과 동일한 연산량을 가진다.

<83> 상기 도 5에서는 천공된 리드 뮤러 부호, 즉 $(n-t, k)$ 리드 뮤러 부호의 IFHT 수행에 대해서 설명하였으며 다음으로 도 6을 참조하여 반복(repetition)된 리드 뮤러 부호, 즉 $(n+t, k)$,

k) 리드 뮤터 부호, 일 예로 (8, 3) 리드 뮤터 부호가 2비트 반복된 (10, 3) 리드 뮤터 부호를 가지고 IFHT를 수행하는 과정을 설명하기로 한다. 여기서, t는 반복되는 비트수를 나타낸다.

<84> 상기 도 6은 일반적인 반복된 리드 뮤터 부호를 IFHT를 사용하여 복호하는 연관정 복호 장치 내부 구조를 도시한 도면이다. 상기 도 6을 설명하기에 앞서, 먼저, 상기 (10,3) 리드 뮤터 부호는 상기 표 1에서 설명한 (8, 3) 리드 뮤터 부호의 부호어들 각각에서 선행하는 2비트 (2bits)씩을 반복한 것이며, 이를 나타내면 하기 표 6과 같다.

<85> 【표 6】

정보 비트	부호어
000	0000000000
001	0101010101
010	0011001100
011	0110011001
100	0000111100
101	0101101001
110	0011110000
111	0110100101

<86> 그리고, 상기 (10, 3) 리드 뮤터 부호에서 발생 가능한 모든 부호어를 각각을 상기 BPSK 방식으로 변조하였을 때의 변조 성분들을 나타내면 하기 표 7과 같다.

<87> 【표 7】

+	+	+	+	+	+	+	+	+	+	+
+	-	+	-	+	-	+	-	+	-	+
+	+	-	-	+	+	-	-	+	+	+
+	-	-	+	+	-	-	+	+	+	-
+	+	+	+	-	-	-	-	+	+	+
+	-	+	-	+	-	+	-	+	+	-
+	+	-	-	-	-	+	+	+	+	+
+	-	-	+	-	+	+	-	+	-	-

<88> 상기 도 6을 참조하면, 먼저 수신 신호 r 은 (10,3) 리드 뮤터 부호에 잡음 성분 및 간섭 성분이 삽입된 신호이므로 하기와 같이 나타낸다.

<89> $r = r_1 \ r_2 \ r_3 \ r_4 \ r_5 \ r_6 \ r_7 \ r_8 \ r_9 \ r_{10}$

<90> 상기 수신 신호 r 은 누적기(accumulator)(611)로 전달되고, 상기 누적기(611)는 상기 수신 신호 r 을 입력하여 최하위 비트 LSB: Least Significant Bit, 이하 "LSB"라 칭하기로 한다)부터 2비트를 최상위 비트 MSB: Most Significant Bit, 이하 "MSB"라 칭하기로 한다)부터 2비트와 누적한 후 IFHT기(613)로 출력한다. 여기서, 상기 누적기(611)는 송신측에서 (8, 3) 리드 뮤터 부호에서 반복한 위치의 비트들을 누적하는 것이며, 상기 반복 위치 관련 정보는 송신측과 수신측 상호가 인지하고 있다. 상기 IFHT기(613)는 상기 누적기(611)에서 출력한 신호를 입력하여 IFHT를 수행한 후 그 결과를 비교 선택기(615)로 출력한다. 상기 비교 선택기(615)는 상기 IFHT기(613)에서 출력한 모든 IFHT 결과값들을 비교하여 최대 상관값을 가지는 부호어를 선택하여 상기 선택한 부호어를 상기 송신측에서 송신한 부호어로 판단한다. 결과적으로, 상기 비교 선택기(615)에서 출력한 부호어에 상응하는 정보 비트가 원래의 정보 비트로 복원되는 것이다. 상기 IFHT 수행 과정은 상기 종래 기술 부분의 도 4에서 설명하였으므로 여기서는 그 상세한 설명을 생략하기로 한다. 결국, 상기 천공된 $(n+t, k)$ 리드 뮤터 부호에 대해서 IFHT를 사용하여 연판정 복호할 때는 상기 (n, k) 리드 뮤터 부호에서 반복된 위치의 비트들을 누적하여 (n, k) 리드 뮤터 부호화하여 IFHT를 사용하여 연판정 복호하는 것과 동일하게 된다. 또한, 연판정 복호를 위한 연산량 역시 상기 (n, k) 리드 뮤터 부호의 연산량과 동일한 연산량을 가진다.

<91> 상기에서 설명한 바와 같이 리드 뮤터 부호가 천공된 형태일 경우에는 천공된 위치에 0을 삽입하여 IFHT를 수행하고, 상기 리드 뮤터 부호가 반복된 형태일 경우에는 반복된 비트들을 누적하여 IFHT를 수행함으로써 모두 연판정 복호 가능하다. 상기 설명에서는 리드 뮤터 부호에 마스크(mask)를 적용하지 않은 경우를 예로 하였으나, 상기 리드 뮤터 부호에 마스크를

적용할 경우는 상기 종래 기술 부분에서 마스크를 고려할 경우 IFHT를 적용할 경우와 동일하게 동작한다. 본 발명에서는 이런 IFHT 특성들을 이용하여 임의의 정보 비트수와 블록 비트수를 가지는 블록 부호들을 리드.물러 부호가 천공된 형태, 혹은 반복된 형태, 혹은 마스킹된 형태로 가정하여 IFHT를 사용하여 연관정 복호하는 방안을 제공한다.

<92> 도 7은 본 발명의 제1실시예에 따른 IFHT기를 사용한 연관정 복호 장치 내부 구조를 도시한 도면이다.

<93> 상기 도 7을 참조하면, 상기 본 발명의 제1실시예에 따른 연관정 복호 장치는 제어기(controller)(700)와, 마스크 곱셈기(mask multiplier)(710)와, 심볼 배치기(symbol arrange unit)(720)와, IFHT기(730)와, 비교 선택기(740)로 구성된다. 그리고, 상기 본 발명의 제1실시 예 및 하기에서 설명할 본 발명의 제2실시예에서 송수신되는 블록 부호에 적용되는 생성 행렬은 하기 수학식 3과 같다고 가정하기로 한다.

<94>

$$G = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$

【수학식 3】

<95> 상기 생성 행렬은 6×11 행렬이며, 상기 수학식 3과 같은 생성 행렬을 적용할 경우 (11, 6) 블록 부호가 생성된다. 그리고, 상기 생성 행렬 G의 6개의 기저들중 제1행(row)부터 제4행 까지의 상위 4개의 기저들을 IFHT 수행을 위한 IFHT 입력 목적으로 사용하고, 상기 상위 4개의 기저들을 제외한 나머지 하위 2개의 기저들, 즉 제5행 및 제6행의 기저들을 마스크 기저(mask basis)로 사용한다고 가정하기로 한다. 따라서, 상기 생성 행렬에서 4개의 기저들만을 IFHT 입력 목적으로 사용하기 때문에, 상기 (11, 6) 블록 부호를 상관하기 위한 IFHT의 입력 크기는

$2^4 = 16$ 이 된다. 상기 본 발명의 제1실시예에서는 송신측과 수신측이 상기 IFHT 크기 정보 및 생성 행렬 정보를 상호간에 인지하고 있다.

<96> 그리고, 상기에서 설명한 바와 같이 송신측에서 송신한 (11, 6) 블록 부호는 잡음 성분과 간섭 성분이 포함된 수신 신호 r 로 수신측에 수신되며, 상기 수신 신호 r 은 다음과 같이 표현하기로 한다.

<97> $r = r_1 \ r_2 \ r_3 \ r_4 \ r_5 \ r_6 \ r_7 \ r_8 \ r_9 \ r_{10} \ r_{11}$

<98> 여기서, 상기 r_1 내지 r_{11} 각각을 수신 심볼이라 칭하기로 한다. 상기 수신 신호 r 은 마스크 곱셈기(710)로 전달되고, 상기 마스크 곱셈기(710)는 상기 수신 신호 r 과 제어기(700)에서 출력하는 마스크($mask$) M_i 를 곱한 후 심볼 배치기(720)로 출력한다. 상기 제어기(700)는 상기 수신측의 주제어기(main controller, 도시하지 않음)로부터 생성 행렬에 관련된 생성 행렬 정보와, IFHT 크기 정보를 전달받고, 상기 전달받은 생성 행렬 정보와, IFHT 크기 정보를 가지고 마스크 M_i 와 심볼 위치 정보를 생성한다. 상기 제어기(700)는 상기 생성한 마스크 M_i 는 상기 마스크 곱셈기(710)로 출력하고, 상기 생성한 심볼 위치 정보는 상기 심볼 배치기(720)로 출력한다. 여기서, 상기 제어기(700)는 상기 생성 행렬에서 하위 2개의 기저들을 마스크 기저로 사용한다고 가정했으므로 상기 생성 행렬의 제5행과 제6행을 각각 마스크 기저 m_1 과 m_2 로 정의한다. 그러면, 상기 마스크로 사용되는 마스크 기저는 m_1 기저와, m_2 기저와, m_1 기저와 m_2 기저의 배타적 논리합(XOR)(이하 " $m_1 \oplus m_2$ "라 칭하기로 한다)이다.

<99> 또한, 상기 본 발명의 제1실시예에 따른 연관정 복호 장치는 직렬(serial) 구조를 가지기 때문에, 실제 마스크가 적용되지 않았을 경우의 하드웨어(hardware) 동작을 마스크가 적용될 경우와 동일하게 고려하기 위해서 all-one 마스크를 추가

로 적용한다. 즉, 상기 제어기(700)는 상기 마스크 곱셈기(710)로 all-one 마스크를 출력함으로써 실제 마스크가 적용되지 않은 경우 상기 마스크가 적용된 경우와 동일한 하드웨어 구조를 가지고도 마스크가 적용되지 않은 경우까지 고려하는 것을 가능하게 한다. 상기 m_1 마스크 기저와, m_2 마스크 기저와, m_1 마스크 기저와 m_2 기저의 배타적 논리합 마스크 기저 $m_1 \oplus m_2$ 각각은 BPSK 방식으로 변조되어 마스크로 사용되며, 상기 종래 기술 부분에서 설명한 바와 같이 디지털 데이터(digital data)는 상기 BPSK 방식으로 변조되기 때문에 상기 디지털 데이터 "0"은 "+1"로, 상기 디지털 데이터 "1"은 "-1"로 대응된다. 또한, 상기 종래 기술 부분에서 설명한 바와 마찬가지로 상기 제어기(700)는 상기 마스크가 적용되지 않았을 경우 상기 all-one 마스크를 출력하지 않고 상기 마스크 곱셈기(710)의 동작을 그대로 바이패스(bypass)할 수도 있음을 물론이다.

<100> 한편, 상기 종래 기술 부분에서 설명한 바와 같이 IFHT기를 사용하는 연판정 복호 장치는 직렬 구조 혹은 병렬(parallel) 구조를 가지는데, 상기 직렬 구조라 함은 하기에서 설명할 마스크 M_i 를 순차적으로 고려하는 형태이며, 상기 병렬 구조라 함은 상기 마스크 M_i 를 일시적으로 병렬 고려하는 형태를 나타낸다. 본 발명에서는 설명의 편의상 상기 직렬 구조의 연판정 복호 장치만을 설명하기로 하며, 본 발명이 병렬 구조의 연판정 복호 장치에도 적용 가능함은 물론이다.

<101> 상기 심볼 배치기(720)는 상기 제어기(700)에서 출력하는 심볼 위치 정보에 상응하게 상기 마스크 곱셈기(710)에서 출력된 신호, 즉 수신 신호 r 을 입력하여 상기 수신 신호 r 을 구성하는 심볼들의 위치를 재배치하여 상기 IFHT기(730)로 출력한다. 여기서, 상기 제어기(700)의 심볼 위치 정보 결정 과정 및 상기 심볼 배치기(720)의 심볼 재배치 과정은 하기에서 설명할

것이므로 여기서는 그 상세한 설명을 생략하기로 한다. 한편, 상기 주제어기는 상기 IFHT 크기 정보를 상기 제어기(700) 뿐만 아니라 상기 IFHT기(730)로도 전달한다. 상기 IFHT기(730)는 상기 주제어기로부터 전달받은 IFHT 크기 정보에 따라 IFHT의 입력 및 해당 스테이지(stage)를 가지는 IFHT를 구성하고, 상기 심볼 배치기(720)에서 출력된 신호를 IFHT 수행하여 그 결과를 상기 비교 선택기(740)로 출력한다. 상기 비교 선택기(740)는 상기 IFHT기(730)에서 출력한 모든 IFHT 결과값들을 비교하여 최대 상관값을 가지는 부호어를 선택하여 상기 선택한 부호어를 상기 송신측에서 전송한 부호어로 판단한다. 결과적으로, 상기 비교 선택기(730)에서 출력한 부호어에 상응하는 정보 비트가 원래의 정보 비트로 복원되는 것이다.

<102> 그러면 여기서 도 8을 참조하여 상기 제어기(700)의 심볼 위치 정보 결정 과정 및 심볼 배치기(720)의 수신 심볼 재배치 과정을 설명하기로 한다.

<103> 상기 도 8은 도 7의 제어기(700)의 심볼 위치 정보 결정 과정 및 심볼 배치기(720)의 수신 심볼 재배치 과정을 개략적으로 도시한 도면이다.

<104> 상기 도 8을 참조하면, 먼저 상기에서 수학식 3의 생성 행렬에서 제1행 내지 제4행의 상위 4개의 기저들만 IFHT 입력을 위해 사용된다고 가정하였으므로, IFHT기(730)의 입력은 2⁴개 (=16개)가 필요하다. 여기서, 상기 IFHT기(730)의 입력은 0부터 15까지 16개이며, 하기의 설명에서 설명의 편의상 상기 0에 해당하는 입력을 "0번째 입력", 15에 해당하는 입력을 "15번째 입력"이라 칭하기로 하며, 실질적으로는 상기 "0번째 입력"이 상기 IFHT기(730)의 제1입력이 되며, 상기 "15번째 입력"이 상기 IFHT기(730)의 제16입력이 된다. 그러면, 상기 제어기(700)가 상기 수신 신호 r, 즉 r₁ r₂ r₃ r₄ r₅ r₆ r₇ r₈ r₉ r₁₀ r₁₁을 상기 IFHT기(730)의 입력으로 고려하기 위해 상기 수신 신호 r을 구성하는 심볼들 각각, 즉 r₁, r₂, r₃, r₄, r₅, r₆, r₇, r₈, r₉, r₁₀, r₁₁ 각각의 심볼 위치 정보를 결정하는 과정을 설명하면 다음과 같다.

<105> 상기 제어기(700)는 상기 수신 신호 r의 심볼들 각각을 순서대로, 즉 r1부터 순차적으로 r11까지 상기 IFHT기(730)의 입력으로 대응시키기 위한 심볼 위치를 결정한다. 첫 번째로, 상기 제어기(700)(720)는 상기 생성 행렬의 제1열(column)의 상위 4비트들만을 고려하여 제1행의 엘리먼트를 LSB로, 제4행의 엘리먼트를 MSB로 하여 "0011"을 생성하고, 상기 "0011"을 십진수로 변환한다. 상기 "0011"을 십진수로 변환하면 "3"이 되므로, 상기 제어기(700)는 2⁴개 (=16개), 즉 0 내지 15까지의 입력 크기 16인 상기 IFHT기(730)의 3번째 입력으로 상기 수신 신호 r의 첫 번째 심볼 r1이 배치될 수 있도록 심볼 위치를 결정한다. 두 번째로, 상기 제어기(700)는 상기 생성 행렬의 제2열의 상위 4비트들만을 고려하여 제1행의 엘리먼트를 LSB로, 제4행의 엘리먼트를 MSB로 하여 "1100"을 생성하고, 상기 "1100"을 십진수로 변환한다. 상기 "1100"을 십진수로 변환하면 "8"이 되므로, 상기 제어기(700)는 상기 IFHT기(730)의 8번째 입력으로 상기 수신 신호 r의 두 번째 심볼 r2이 배치될 수 있도록 심볼 위치를 결정한다. 세 번째로, 상기 제어기(700)는 상기 생성 행렬의 제3열의 상위 4비트들만을 고려하여 제1행의 엘리먼트를 LSB로, 제4행의 엘리먼트를 MSB로 하여 "0110"을 생성하고, 상기 "0110"을 십진수로 변환한다. 상기 "0110"을 십진수로 변환하면 "6"이 되므로, 상기 제어기(700)는 상기 IFHT기(730)의 6번째 입력으로 상기 수신 신호 r의 세 번째 심볼 r3이 배치될 수 있도록 심볼 위치를 결정한다. 상기 제어기(700)는 이런 식으로 상기 수신 신호 r의 열한번째 심볼 r11까지 상기 IFHT기(730)의 입력으로 배치될 수 있도록 심볼 위치를 결정한다.

<106> 상기 제어기(700)는 상기 도 8에 나타낸 바와 같이 상기 수신 신호 r의 첫 번째 심볼 r1은 상기 IFHT기(730)의 3번째 입력으로, 두 번째 심볼 r2은 상기 IFHT기(730)의 8번째 입력으로, 세 번째 심볼 r3은 상기 IFHT기(730)의 6번째 입력으로, 네 번째 심볼 r4은 상기 IFHT기(730)의 1번째 입력으로, 다섯 번째 심볼 r5은 상기 IFHT기(730)의 12번째 입력으로, 여섯 번

째 심볼 r6은 상기 IFHT기(730)의 6번째 입력으로, 일곱 번째 심볼 r7은 상기 IFHT기(730)의 8번째 입력으로, 여덟 번째 심볼 r1은 상기 IFHT기(730)의 8번째 입력으로, 아홉 번째 심볼 r9은 상기 IFHT기(730)의 9번째 입력으로, 열 번째 심볼 r10은 상기 IFHT기(730)의 8번째 입력으로, 열한 번째 심볼 r11은 상기 IFHT기(730)의 3번째 입력으로 배치될 수 있도록 심볼 위치를 결정하는 것이다.

<107> 특히, 상기 수신 신호 r의 첫 번째 심볼 r1과 열한 번째 심볼 r11은 동일한 십진수 값, 즉 3이 나왔으므로 상기 첫 번째 심볼 r1과 열한 번째 심볼 r11이 가산되어 상기 IFHT기(730)의 3번째 입력으로 배치될 수 있도록 그 심볼 위치가 결정되며, 상기 수신 신호 r의 세 번째 심볼 r3과 여섯 번째 심볼 r6은 동일한 십진수 값, 즉 6이 나왔으므로 상기 세 번째 심볼 r3과 여섯 번째 심볼 r6이 가산되어 상기 IFHT기(730)의 6번째 입력으로 배치될 수 있도록 그 심볼 위치가 결정되며, 상기 수신 신호 r의 두 번째 심볼 r2와 일곱 번째 심볼 r7와 여덟 번째 심볼 r8과 열 번째 심볼 r10은 동일한 십진수 값, 즉 8이 나왔으므로 상기 두 번째 심볼 r2와 일곱 번째 심볼 r7와 여덟 번째 심볼 r8과 열 번째 심볼 r10이 가산되어 상기 IFHT기(730)의 8번째 입력으로 배치될 수 있도록 심볼 위치가 결정된다.

<108> 상기 제어기(700)는 상기 결정한 심볼 위치에 따라 상기 심볼 위치 정보를 생성하고, 상기 생성한 심볼 위치 정보를 상기 심볼 배치기(720)로 출력한다. 상기 심볼 배치기(720)가 상기 제어기(700)에서 출력한 심볼 위치 정보에 상응하게 수신 신호 r의 수신 심볼들 각각을 상기 IFHT기(730)의 입력으로 재배치한다. 그러면 여기서 상기 제어기(700)가 상기 심볼 위치 정보를 생성하는 과정을 정리하면 다음과 같다.

<109> 먼저, 생성 행렬이

$k \times n$ 행렬이며, 상기 생성 행렬의 n 개의 기저들중 상위 $k - m$ 개의 기저들을 IFHT 수행을 위한 IFHT 입력 목적으로 사용하고, 상기 상위 $k - m$ 개의 기저들을 제외한 나머지 하위 m 개의 기저들을 마스크 기저로 사용한다고 가정하기로 한다. 상기 생성 행렬에 따라 생성된 (n, k) 블록 부호는 채널상에서 잡음 성분과 간섭 성분이 가산되어 수신측으로 수신되며, 상기 수신된 수신 신호 r 은 $r = r_1 \ r_2 \ \dots \ r_{(n-1)} \ r_n$ 으로 표현된다. 상기 제어기(700)는 상기 생성 행렬의 제1열부터 순차적으로 제 n 열까지 각각 상위 $k - m$ 비트들만을 고려하여 제1행의 엘리먼트를 LSB로, 제 $k - m$ 행의 엘리먼트를 MSB로 하여 이진 시퀀스(binary sequence)들을 생성하고, 상기 생성 한 이진 시퀀스들 각각을 십진수로 변환한다. 그리고, 상기 제어기(700)는 입력 크기 2^{k-m} 인 IFHT기(730)의 상기 변환된 십진수들 각각에 상응하는 입력으로 상기 수신 신호 r 의 첫 번째 심볼 r_1 부터 순차적으로 제 n 번째 심볼 r_n 을 배치하도록 심볼 위치를 결정한다. 여기서, 상기 IFHT기(730)의 입력은 0부터 $(2^{k-m}-1)$ 까지 2^{k-m} 개이며, 상기 0에 해당하는 입력을 "0번째 입력", $(2^{k-m}-1)$ 에 해당하는 입력을 " $(2^{k-m}-1)$ 번째 입력"이라 칭하기로 하며, 실질적으로는 상기 "0번째 입력"이 상기 IFHT기(730)의 제1입력이 되며, 상기 " $(2^{k-m}-1)$ 번째 입력"이 상기 IFHT기(730)의 제 2^{k-m} 입력이 된다. 또한, 상기 제어기(700)는 상기 생성 행렬의 i 번째 열과 j 번째 열의 십진수 값이 둘다 "a"라면, 상기 수신 신호의 i 번째 심볼 r_i 와 j 번째 심볼 r_j 를 가산한 $r_i + r_j$ 가 상기 IFHT기(730)의 a 번째 입력으로 배치될 수 있도록 심볼 위치를 결정한다.

<110> 다음으로, 도 9를 참조하여 상기 심볼 배치기(720)의 내부 구조를 설명하기로 한다.

<111> 상기 도 9는 도 7의 심볼 배치기(720)의 내부 구조를 도시한 도면이다.

<112> 먼저, 상기 심볼 배치기(720)는 상기 제어기(700)에서 출력하는 심볼 위치 정보에 상응 하게 상기 마스크 곱셈기(710)에서 출력한 신호의 심볼들 각각을 재배치하여 상기 IFHT기(730)

의 입력으로 제공하는데, 이런 재배치 과정에 상응하는 구조가 상기 도 9에 도시되어 있다. 상기 심볼 배치기(720)는 스위치(switch)(901)와, 2^k 개의 가산기들(911, 921, 931, 941)과, 2^k 개의 메모리(memoer)들(915, 925, 935, 945)과, $.2^k$ 개의 스위치들(915, 925, 935, 945)로 구성된다. 여기서, 상기 2^k 개의 메모리들(915, 925, 935, 945) 각각은 최초에 "0"으로 초기화되어 있다. 상기 도 9를 참조하면, 먼저 입력 신호, 즉 상기 마스크 곱셈기(710)의 출력 신호와 심볼 위치 정보가 상기 심볼 배치기(720)로 입력된다. 이하 설명에서는 설명의 편의상 마스크 곱셈기(710)에서 출력되는 신호가 수신 신호 r 에 all-one 마스크가 적용된 경우(즉, 실제 마스크가 적용되지 않은 경우)를 일 예로 하여 설명하기로 한다. 상기 마스크 곱셈기(710)에서 마스크 M_i 가 적용된 경우 역시 수신 신호 r 에 마스크 값만 곱해졌을 뿐 그 심볼 재배치 과정은 상기 마스크 M_i 가 적용되지 않은 경우와 동일하게 동작함은 물론이다.

<113> 상기 all-one 마스크가 적용된 경우를 가정하였으므로, 상기 마스크 곱셈기(710)에서 출력하는 신호는 수신 신호 r 과 동일하고, 상기 수신 신호 r 은 상기에서 설명한 바와 같이 $r = r_1 \ r_2 \ r_3 \ r_4 \ r_5 \ r_6 \ r_7 \ r_8 \ r_9 \ r_{10} \ r_{11}$ 이다. 상기 심볼 배치기(720)로 상기 수신 신호 r , 즉 $r_1 \ r_2 \ r_3 \ r_4 \ r_5 \ r_6 \ r_7 \ r_8 \ r_9 \ r_{10} \ r_{11}$ 이 입력되면 상기 스위치(901)는 상기 제어기(700)에서 제공한 심볼 위치 정보에 따라 상기 수신 신호 r 의 수신 심볼들 각각을 해당 위치의 가산기들 각각으로 연결시킨다. 일 예로, 상기 도 8에서 설명한 바와 같이 수신 심볼 r_4 의 심볼 위치 정보는 십진수로 1이므로, 상기 스위치(901)는 상기 r_1 을 메모리 M_1 (923) 전단에 위치한 가산기(921)로 연결한다. 이런 식으로 상기 스위치(901)는 상기 수신 심볼들 각각을 해당 메모리의 전단에 위치하는 가산기들 각각으로 연결하게 된다. 상기 2^k 개의 가산기들(911, 921, 931, 941) 각각은 상기 스위치(901)에서 연결되는 신호와, 2^k 개의 메모리들(915, 925, 935, 945) 각각에서 출력하는 신호를 피드백(feedback)하여 가산한 후 상기 2^k 개의 메모리들(915, 925, 935, 945) 각각에서

각으로 출력한다. 여기서, 상기 2^k 개의 메모리들(915, 925, 935, 945) 각각은 신호가 입력될 때마다 피드백 루프(feedback loop)를 통해 입력되는 신호와 기존에 저장하고 있던 신호를 가산하여 새로이 갱신된(update)된 신호를 저장한다. 따라서, 상기 스위치(901)가 상기 2^k 개의 메모리들(915, 925, 935, 945) 각각의 전단에 위치한 2^k 개의 가산기들(911, 921, 931, 941) 각각에 연결되지 않을 때는 새로이 입력되는 신호가 없기 때문에 기존에 저장하고 있던 신호를 유지하게 된다.

<114> 그래서, 상기 수신 신호 r 의 수신 심볼들 각각에 대한 심볼 재배치가 종료되면 상기 심볼 배치기(720)는 상기 메모리 M_0 (913)부터 메모리 $M_{2^{k-1}}$ (943)까지 저장되어 있는 신호들이 순차적으로 상기 IFHT기(730)에 입력되도록 2^k 개의 스위치들(915, 925, 935, 945) 각각의 스위칭 동작을 제어한다. 즉, 상기 메모리 M_0 (913)에 연결된 스위치(915)가 제일 먼저 상기 IFHT기(730)에 연결되고, 순차적으로해서 마지막으로 상기 메모리 $M_{2^{k-1}}$ (943)에 연결된 스위치(945)가 상기 IFHT기(730)에 연결된다. 그러면, 상기 IFHT기(730)는 상기 메모리 M_0 (913)부터 메모리 $M_{2^{k-1}}$ (943)까지 저장되어 있는 신호들을 순차적으로 입력하여 IFHT 수행하게 된다. 한편, 상기 도 9에서는 상기 2^k 개의 메모리들(915, 925, 935, 945) 각각이 2^k 개의 스위치들(915, 925, 935, 945) 각각과 연결되는 구조를 설명하고 있으나, 상기 2^k 개의 스위치들(915, 925, 935, 945) 대신 병렬/직렬 변환기(P/S convertor)를 사용하여도 된다. 즉, 상기 병렬/직렬 변환기는 상기 2^k 개의 메모리들(915, 925, 935, 945) 각각에서 출력하는 2^k 개의 병렬 입력들을 메모리 M_0 (913)의 출력신호가 최선에 위치하도록 하게 직렬 변환하여 상기 IFHT기(730)에 출력하면 되는 것이다.

<115> 또한, 상기 IFHT기(730)는 최대 2^k 개의 입력들을 가지며,

$h \leq k$ 인 h 에 대해 2^h 개의 입력을 사용하여 h 개의 스테이지 연산을 수행한다. 여기서, 상기 IFHT기(730)의 입력 개수는 미리 결정되어 있을 수도 있고 상황에 적응적으로 가변할 수도 있는데, 상기 IFHT기(730)는 연산량을 고려하여 입력 개수를 결정한다. 즉, 상기에서 설명한 바와 같이 상기 IFHT기(730)는 (n, k) 블록 부호에 대한 복호 과정에서 $2^h \log_2 2^h$ 의 덧셈 과정의 연산량을 필요로 하며, 이런 점들을 고려하여 결국 입력수가 최소일수록 최소 연산량을 가지게 되므로 상기 IFHT기(730)는 최소 연산량을 가지는 입력수를 가변적으로 결정한다. 여기서, 상기 IFHT기(730)가 입력수를 결정하는 과정을 설명하면 다음과 같다.

<116> 먼저, 상기 IFHT기(730)의 입력 개수 2^h 이 결정되어지면, 상기 복호기에서 사용되어지는 마스크(mask) 함수의 개수는 2^{k-h} 개가 되어지고, 따라서, 전체적인 복호기의 연산량이 결정되어진다. 이 때, 모든 $h (0 \leq h \leq k)$ 에 대해서 복호기의 연산량을 계산하고, 이중 최소 연산량을 가지는 h 값을 찾고, IFHT기(730)의 입력 개수를 2^h 으로 결정한다. 먼저, 임의의 변수 h 값에 따른 전체 복호기의 계산량을 생각해보기로 한다, 전체 복호기의 연산량은 마스크 함수가 곱해지는 부분과 IFHT를 수행하는 부분이 거의 대부분을 차지한다. 먼저, n 비트의 길이를 가지는 하나의 입력신호와 하나의 마스크 곱셈에 대한 연산량을 보면, n 번 곱셈과 $n-1$ 번의 덧셈을 필요로 한다. 상기 가정에 따르면, h 값에 따라서 2^{k-h} 개의 마스크들이 사용되어지게 되는데, 이 때 마스크가 곱해기는데 사용되어지는 총 연산량은 $2^{k-h} \times n$ 의 곱셈 과정과, $2^{k-h} \times (n-1)$ 의 덧셈 연산이다. 또한, 하나의 IFHT기의 연산량은 $2^h \log_2 2^h = h \cdot 2^h$ 개의 덧셈이 필요하고, 전체 복호기에서 IFHT연산의 회수는 mask의 개수만큼 진행되므로, 전체 복호기에서 IFHT연산을 위한 연산량은 $h \cdot 2^h \times 2^{k-h} = h \cdot 2^k$ 이 된다. 따라서, 상기 변수 h 값에 따른 복호기의 전체 연산량은 $2^{k-h} \times n$ 번의 곱셈 과정과 $(n-1) \cdot 2^{k-h} + h \cdot 2^k$ 의 덧셈 과정이 되며, 상기 곱셈 과정과 덧셈 과정의 연산 복잡도가 동일하다고 가정할 경우 전체 연산량은

$n \cdot 2^{k-h} + (n-1) \cdot 2^{k-h+h} \cdot 2^k = (2n-1) \cdot 2^{k-h+h} \cdot 2^k$ 이 된다. 따라서, 상기 복호기의 IFHT기의 입력 개수를 결정함에 있어서, 상기 IFHT기(730)는 가능한 모든 $h (0 \leq h \leq k)$ 값에 따른 모든 연산량 값을 계산하여 가장 작은 연산량을 가지는 h 을 택하여 이에 따른 IFHT의 입력 개수, 즉 크기를 결정한다.

<117> 상기 본 발명의 제1실시예에서는 연판정 복호를 위한 IFHT 크기 정보와 생성 행렬 정보를 주제어기에서 제공하는 경우의 연판정 복호를 설명하였으며, 다음으로 본 발명의 제2실시예에서는 생성 행렬 정보만을 주제어기에서 제공하는 경우의 연판정 복호를 설명하기로 한다.

<118> 도 10은 본 발명의 제2실시예에 따른 IFHT기를 사용한 연판정 복호 장치 내부 구조를 도시한 도면이다.

<119> 상기 도 10을 참조하면, 상기 본 발명의 제2실시예에 따른 연판정 복호 장치는 제어기(1000)와, 마스크 곱셈기(1010)와, 심볼 배치기(1020)와, IFHT기(1030)와, 비교 선택기(1040)로 구성된다. 그리고, 상기 본 발명의 제2실시예에서 송수신되는 블록 부호에 적용되는 생성 행렬은 상기 수학식 3과 같다고 가정하기로 한다. 상기 수학식 3의 생성 행렬은

6×11 행렬이며, 상기 수학식 3과 같은 생성 행렬을 적용할 경우 (11, 6) 블록 부호가 생성된다. 상기 제어기(1000)는 상기 수신측의 주제어기(도시하지 않음)로부터 상기 생성 행렬에 관련된 생성 행렬 정보를 전달받고, 상기 전달받은 생성 행렬 정보를 가지고 마스크 M_i 와 심볼 위치 정보 및 IFHT 크기 정보를 생성한다. 결과적으로, 본 발명의 제1실시예와 제2실시예의 차이점은 연판정 복호 장치의 제어기가 어떤 동작을 수행하느냐에 있으며, 결과적으로 본 발명의 제1실시예에서는 제어기(700)가 상기 도 7에서 설명한 바와 같이 주제어기로부터 생성 행렬 정보와 IFHT 크기 정보를 전달받아 마스크 M_i 와 심볼 위치 정보를 결정하는 동작을 수행하며, 본 발명의 제2실시예에서는 제어기(1000)가 주제어기로부터 생성 행렬 정보만을 전달받아 마스크 M_i 와 심볼 위치 정보 및 IFHT 크기 정보를 생성하는 동작을 수행한다는 점에서 차이점이 있다.

<120> 그러면, 여기서 상기 제어기(1000)의 동작을 설명하기로 한다.

<121> 상기 제어기(1000)는 상기 주제어기로부터 전달받은 생성 행렬 정보를 가지고 IFHT기(1030)의 입력 개수를 결정하고, 즉 상기 IFHT기(1030)의 입력으로 사용할 기저들의 개수를 결정한다. 여기서, 상기 제어기(1000)가 IFHT기(1030)의 입력 개수를 결정하는 과정을 설명하면 다음과 같다.

<122> 먼저, 상기 IFHT기(1030)의 입력 개수 2^h 이 결정되어지면, 상기 복호기에서 사용되어지는 마스크(mask) 함수의 개수는 2^{k-h} 개가 되어지고, 따라서, 전체적인 복호기의 연산량이 결정되어진다. 이 때, 모든 $h (0 \leq h \leq k)$ 에 대해서 복호기의 연산량을 계산하고, 이중 최소 연산량을 가지는 h 값을 찾고, IFHT기(1030)의 입력 개수를 2^h 으로 결정한다. 먼저, 임의의 변수 h 값에 따른 전체 복호기의 계산량을 생각해보

기로 한다. 전체 복호기의 연산량은 마스크 함수가 곱해지는 부분과 IFHT를 수행하는 부분이 거의 대부분을 차지한다. 먼저, n 비트의 길이를 가지는 하나의 입력신호와 하나의 마스크 곱셈에 대한 연산량을 보면, n 번 곱셈과 $n-1$ 번의 덧셈을 필요로 한다. 상기 가정에 따르면, h 값에 따라서 2^{k-h} 개의 마스크들이 사용되어지게 되는데, 이 때 마스크가 곱해기는데 사용되어지는 총 연산량은 $2^{k-h} \times n$ 의 곱셈 과정과, $2^{k-h} \times (n-1)$ 의 덧셈 연산이다. 또한, 하나의 IFHT기의 연산량은 $2^h \log_2 2^h = h \cdot 2^h$ 개의 덧셈이 필요하고, 전체 복호기에서 IFHT연산의 회수는 mask의 개수만큼 진행되므로, 전체 복호기에서 IFHT연산을 위한 연산량은 $h \cdot 2^h \times 2^{k-h} = h \cdot 2^k$ 이 된다. 따라서, 상기 변수 h 값에 따른 복호기의 전체 연산량은 $2^{k-h} \times n$ 번의 곱셈 과정과 $(n-1) \cdot 2^{k-h} + h \cdot 2^k$ 의 덧셈 과정이 되며, 상기 곱셈 과정과 덧셈 과정의 연산 복잡도가 동일하다고 가정할 경우 전체 연산량은 $n \cdot 2^{k-h} + (n-1) \cdot 2^{k-h} + h \cdot 2^k = (2n-1) \cdot 2^{k-h} + h \cdot 2^k$ 이 된다. 따라서, 상기 복호기의 IFHT기의 입력 개수를 결정함에 있어서, 상기 IFHT기(1030)는 가능한 모든 h ($0 \leq h \leq k$) 값에 따른 모든 연산량 값을 계산하여 가장 작은 연산량을 가지는 h 을 택하여 이에 따른 IFHT의 입력 개수, 즉 크기를 결정한다.

<123> 상기 제어기(1000)는 IFHT를 수행할 경우의 연산량과, 시스템의 복잡도(complexity)와, IFHT 수행 시간등을 고려하여 상기 IFHT의 입력 개수를 결정한다. 즉, 상기 제어기(100)는 최소의 연산량과, 최소의 시스템 복잡도와, 최소의 IFHT 수행 시간을 가지는 개수로 상기 IFHT의 입력 개수를 결정한다. 일 예로, 상기 생성 행렬이 $k \times n$ 행렬일 경우, 상기 $k \times n$ 생성 행렬로부터 (n, k) 블록 부호가 생성된다. 상기 $k \times n$ 생성 행렬의 k 개의 기저들중 1개의 기저들을 IFHT의 입력 목적으로 사용하고, $k-1$ 개의 기저들을 마스크 기저들로 사용한다면, 상기 IFHT는 2^k 개의 입력에 대해 1개의 스테이지들의 연산을 수행하고, 총 2^{k-1} 개의 마스크 기저들에 대해 상기 연산정 복호가 반복 수행된다.

<124> 상기 도 10에서는 상기 제어기(1000)가 상기 수학식 3의 생성 행렬의 6개의 기저들 중 제1행부터 제3행까지의 상위 3개의 기저들을 IFHT 수행을 위한 IFHT 입력 목적으로 사용하고, 상기 상위 3개의 기저들을 제외한 나머지 하위 3개의 기저들, 즉 제4행 내지 제6행의 기저들을 마스크 기저로 사용하기로 결정하였다고 가정하기로 한다. 상기 제어기(1000)는 상기 결정한 마스크 기저들에 상응하는 마스크 M_i 를 상기 마스크 곱셈기(1010)로 출력하고, 상기 결정한 IFHT의 입력 개수들에 상응하여 수신 신호 r 의 수신 심볼들 각각의 심볼 위치 정보를 결정하여 상기 심볼 배치기(1020)로 출력하고, 상기 IFHT 크기 정보를 상기 심볼 배치기(1020) 및 IFHT 기(1030)로 출력한다.

<125> 여기서, 상기 제어기(1000)는 상기 생성 행렬 G 에서 하위 3개의 기저들을 마스크 기저로 사용한다고 가정했으므로 상기 생성 행렬 G 의 제4행 내지 제6행을 각각 제1 마스크 m_1 내지 제3 마스크 m_3 로 정의한다. 그러면, 상기 마스크로 사용되는 마스크 기저는 m_1 기저와, m_2 기저와, m_1 기저와 m_2 기저의 배타적 논리합(이하 " $m_1 \oplus m_2$ "라 칭하기로 한다)과, m_1 기저와 m_3 기저의 배타적 논리합(이하 " $m_1 \oplus m_3$ "라 칭하기로 한다)과, m_2 기저와 m_3 기저의 배타적 논리합(이하 " $m_2 \oplus m_3$ "라 칭하기로 한다)과, m_1 기저와 m_2 기저와 m_3 기저의 배타적 논리합(이하 " $m_1 \oplus m_2 \oplus m_3$ "라 칭하기로 한다)이다.

<126> 또한, 상기 본 발명의 제2실시예에 따른 연판정 복호 장치는 직렬 구조를 가지기 때문에 실제 마스크가 적용되지 않았을 경우의 하드웨어 동작을 마스크가 적용될 경우와 동일하게 고려하기 위해서 all-one 마스크를 추가로 적용한다. 즉, 상기 제어기(1000)는 상기 마스크 곱셈기(1010)로 all-one 마스크를 출력함으로써 실제 마스크가 적용되지 않은 경우 상기 마스크가 적용된 경우와 동일한 하드웨어 구조를 가지고도 마스크가 적용되지 않은 경우까지 고려하는 것을 가능하게 한다. 상기 m_1 마스크 기저와, m_2 마스크 기저와, m_1 마스크 기저와 m_2 기저

의 배타적 논리합 마스크 기저 $m_1 \oplus m_2$ 와, m_1 기저와 m_3 기저의 배타적 논리합 마스크 기저 $m_1 \oplus m_3$ 와, m_2 기저와 m_3 기저의 배타적 논리합 마스크 기저 $m_2 \oplus m_3$ 와, m_1 기저와 m_2 기저와 m_3 기저의 배타적 논리합 마스크 기저 $m_1 \oplus m_2 \oplus m_3$ 각각은 BPSK 방식으로 변조되어 마스크로 사용되며, 상기 종래 기술 부분에서 설명한 바와 같이 디지털 데이터는 상기 BPSK 방식으로 변조되기 때문에 상기 디지털 데이터 "0"은 "+1"로, 상기 디지털 데이터 "1"은 "-1"로 대응된다. 또한, 상기 종래 기술 부분에서 설명한 바와 마찬가지로 상기 제어기(1000)는 상기 마스크가 적용되지 않았을 경우 상기 all-one 마스크를 출력하지 않고 상기 마스크 곱셈기(1010)의 동작을 그대로 바이패스(bypass)할 수도 있음을 물론이다.

<127> 한편, 상기 종래 기술 부분에서 설명한 바와 같이 IFHT기를 사용하는 연판정 복호 장치는 직렬 구조 혹은 병렬 구조를 가지는데, 본 발명에서는 설명의 편의상 상기 직렬 구조의 연판정 복호 장치만을 설명하기로 하였으므로, 상기 도 10의 연판정 복호 장치 역시 직렬 구조를 가진다. 그러면 상기 제어기(1000)는 상기 수신 신호 r 이 수신되면 최초에는 all-one 마스크가 적용되었다고 가정하고 상기 all-one 마스크를 마스크 곱셈기(1010)로 출력한다. 상기 마스크 곱셈기(1010)는 상기 수신 신호 r 과 상기 all-one 마스크를 곱한 후 상기 심볼 배치기(1020)로 출력한다. 상기 심볼 배치기(1020)는 상기 마스크 곱셈기(1010)에서 출력된 신호, 즉 수신 신호 r 을 입력하여 상기 수신 신호 r 을 구성하는 심볼들의 위치를 상기 제어기(1000)에서 제공한 심볼 위치 정보에 상응하게 재배치하여 상기 IFHT기(1030)로 출력한다. 상기 IFHT기(1030)는 상기 제어기(1000)로부터 전달받은 IFHT 크기 정보에 따라 IFHT의 입력 및 해당 스테이지를 가지는 IFHT를 구성하고, 상기 심볼 배치기(1020)에서 출력된 신호를 IFHT 수행하여 그 결과를 상기 비교 선택기(1040)로 출력한다. 상기 비교 선택기(1040)는 상기 IFHT기(1030)에서 출력한 모든 IFHT 결과값들을 비교하여 최대 상관값을 가지는 부호어를 선택하여 상기 선택

한 부호어를 상기 송신측에서 전송한 부호어로 판단한다. 결과적으로, 상기 비교 선택기(1030)에서 출력한 부호어에 상응하는 정보 비트가 원래의 정보 비트로 복원되는 것이다.

<128> 그러면 여기서 도 11을 참조하여 상기 제어기(1000)의 심볼 위치 정보 결정 과정 및 상기 심볼 배치기(1020)의 심볼 재배치 과정을 설명하기로 한다.

<129> 상기 도 11은 도 10의 제어기(1000)의 심볼 위치 정보 결정 과정 및 심볼 배치기(1020)의 수신 심볼 재배치 과정을 개략적으로 도시한 도면이다.

<130> 상기 도 11을 참조하면, 먼저 상기 제어기(1000)가 상기 수학식 3의 생성 행렬에서 제1행 내지 제3행의 상위 3개의 기저들만 IFHT기의 입력을 위해 사용하기로 결정하였으므로, IFHT기(1030)의 입력은 2³개(=8개)가 필요하다. 여기서, 상기 IFHT기(1030)의 입력은 0부터 7까지 8개이며, 하기의 설명에서 설명의 편의상 상기 0에 해당하는 입력을 "0번째 입력", 7에 해당하는 입력을 "7번째 입력"이라 칭하기로 하며, 실질적으로는 상기 "0번째 입력"이 상기 IFHT기(1030)의 제1입력이 되며, 상기 "7번째 입력"이 상기 IFHT기(1030)의 제8입력이 된다. 그러면, 상기 제어기(1000)가 상기 수신 신호 r, 즉 r1 r2 r3 r4 r5 r6 r7 r8 r9 r10 r11을 상기 IFHT기(1030)의 입력으로 고려하기 위해 상기 수신 신호 r을 구성하는 심볼들 각각, 즉 r1, r2, r3, r4, r5, r6, r7, r8, r9, r10, r11 각각의 심볼 위치 정보를 결정하는 과정을 설명하면 다음과 같다.

<131> 상기 제어기(1000)는 상기 수신 신호 r의 심볼들 각각을 순서대로, 즉 r1부터 순차적으로 r11까지 상기 IFHT기(1030)의 입력으로 대응시키기 위한 심볼 위치를 결정한다. 첫 번째로, 상기 제어기(1000)는 상기 생성 행렬 G의 제1열의 상위 3비트들만을 고려하여 제1행의 엘리먼트를 LSB로, 제3행의 엘리먼트를 MSB로 하여

"011"을 생성하고, 상기 "011"을 십진수로 변환한다. 상기 "011"을 십진수로 변환하면 "3"이 되므로, 상기 제어기(1000)는 2^3 개(=8개), 즉 0 내지 7까지의 입력 크기 8인 상기 IFHT기(1030)의 3번째 입력으로 상기 수신 신호 r 의 첫 번째 심볼 r_1 이 배치될 수 있도록 심볼 위치를 결정한다. 두 번째로, 상기 제어기(1000)는 상기 생성 행렬 G 의 제2열의 상위 3비트들만을 고려하여 제1행의 엘리먼트를 LSB로, 제3행의 엘리먼트를 MSB로 하여 "000"을 생성하고, 상기 "000"을 십진수로 변환한다. 상기 "000"을 십진수로 변환하면 "0"이 되므로, 상기 제어기(1000)는 상기 IFHT기(1030)의 0번째 입력으로 상기 수신 신호 r 의 두 번째 심볼 r_2 가 배치될 수 있도록 심볼 위치를 결정한다. 세 번째로, 상기 제어기(1000)는 상기 생성 행렬 G 의 제3열의 상위 3비트들만을 고려하여 제1행의 엘리먼트를 LSB로, 제3행의 엘리먼트를 MSB로 하여 "110"을 생성하고, 상기 "110"을 십진수로 변환한다. 상기 "110"을 십진수로 변환하면 "6"이 되므로, 상기 제어기(1000)는 상기 IFHT기(1030)의 6번째 입력으로 상기 수신 신호 r 의 세 번째 심볼 r_3 이 배치될 수 있도록 심볼 위치를 결정한다. 상기 제어기(1000)는 이런 식으로 상기 수신 신호 r 의 열한번째 심볼 r_{11} 까지 상기 IFHT기(1030)의 입력으로 배치될 수 있도록 심볼 위치를 결정한다. 상기 도 11에 나타낸 바와 같이 상기 제어기(1000)는 상기 수신 신호 r 의 첫 번째 심볼 r_1 은 상기 IFHT기(1030)의 3번째 입력으로, 두 번째 심볼 r_2 은 상기 IFHT기(1030)의 0번째 입력으로, 세 번째 심볼 r_3 은 상기 IFHT기(1030)의 6번째 입력으로, 네 번째 심볼 r_4 은 상기 IFHT기(1030)의 1번째 입력으로, 다섯 번째 심볼 r_5 은 상기 IFHT기(1030)의 4번째 입력으로, 여섯 번째 심볼 r_6 은 상기 IFHT기(1030)의 6번째 입력으로, 일곱 번째 심볼 r_7 은 상기 IFHT기(1030)의 0번째 입력으로, 여덟 번째 심볼 r_8 은 상기 IFHT기(1030)의 0번째 입력으로, 아홉 번째 심볼 r_9 은 상기 IFHT기(1030)의 1번째 입력으로, 열 번째 심볼 r_{10} 은 상기 IFHT기(1030)의 0번째 입력으로, 열한 번째 심볼 r_{11} 은 상기 IFHT기(1030)의 3번째 입력으로 배치될 수 있도록 심볼 위

치를 결정하고, 상기 결정된 심볼 위치에 따른 심볼 위치 정보를 상기 심볼 배치기(1020)로 출력하는 것이다.

<132> 특히, 상기 수신 신호 r의 첫 번째 심볼 r2과 일곱 번째 심볼 r7과 여덟 번째 심볼 r8과 열 번째 심볼 r10은 동일한 십진수 값, 즉 0이 나왔으므로 상기 첫 번째 심볼 r2과 일곱 번째 심볼 r7과 여덟 번째 심볼 r8과 열 번째 심볼 r10이 가산되어 상기 IFHT기(1030)의 0번째 입력으로 배치될 수 있도록 심볼 위치가 결정되며, 상기 수신 신호 r의 네 번째 심볼 r4와 아홉 번째 심볼 r9은 동일한 십진수 값, 즉 1이 나왔으므로 상기 네 번째 심볼 r4와 아홉 번째 심볼 r9이 가산되어 상기 IFHT기(1030)의 1번째 입력으로 배치될 수 있도록 심볼 위치가 결정되며, 상기 수신 신호 r의 첫 번째 심볼 r1와 열한 번째 심볼 r11은 동일한 십진수 값, 즉 3이 나왔으므로 상기 첫 번째 심볼 r1와 열한 번째 심볼 r11이 가산되어 상기 IFHT기(1030)의 3번째 입력으로 배치될 수 있도록 심볼 위치가 결정되며, 상기 수신 신호 r의 세 번째 수신 심볼 r3와 여섯 번째 수신 심볼 r6은 동일한 십진수 값, 즉 6이 나왔으므로 상기 세 번째 수신 심볼 r3와 여섯 번째 수신 심볼 r6이 가산되어 상기 IFHT기(1030)의 6번째 입력으로 배치될 수 있도록 심볼 위치가 결정된다.

<133> 또한, 상기 본 발명의 제2실시예에 따른 심볼 배치기(1020) 구조는 상기 도 9에서 설명한 바와 동일한 구조를 가지며, 다만 IFHT기(1030)의 입력 개수가 상이하므로, 상기 IFHT기(1030)의 전단에 연결되는 메모리들의 개수와, 상기 메모리들에 연결되는 가산기들의 개수 및 상기 메모리들에 연결되는 스위치들의 개수만이 상이해질 뿐이다.

【발명의 효과】

<134> 상술한 바와 같은 본 발명은, 통신 시스템에서 임의의 정보 비트 길이를 가지고 생성된 임의의 블록 길이를 가지는 블록 부호를 심볼 재배치를 통해 IFHT를 수행할 수 있도록 제어함으로써 최소의 연산량을 가지는 연관정 복호를 가능하게 한다는 이점을 가진다. 또한, 상기 블록 부호의 생성 행렬에 따라 IFHT의 입력으로 사용될 기저와 마스크로 사용될 기저의 개수를 결정함으로써 최소의 연상량과, 최소의 시스템 복잡도 및 최소의 IFHT 수행 시간을 가지는 연관정 복호를 가능하게 한다는 이점을 가진다. 또한, 상기와 같이 임의의 정보 비트 길이를 가지고 생성된 임의의 블록 길이를 가지는 블록 부호에 대한 복호를 생성 행렬에 상응하게 제어하여 연관정 복호함으로써 하드웨어적으로 동일한 연관정 복호 장치를 가지고 복호하는 것이 가능하다는 이점을 가진다.

【특허청구범위】**【청구항 1】**

k 개의 정보 비트들과 n 개의 출력 비트들을 가지는 (n, k) 블록 부호를 복호하는 장치에 있어서,

상기 (n, k) 블록 부호의 생성 행렬 정보와, 상기 (n, k) 블록 부호를 역 고속 하다마드 변환(IFHT: Inverse Fast Hadamard Transform)하기 위한 IFHT 크기 정보를 가지고 상기 (n, k) 블록 부호를 구성하는 부호화 심볼들 각각을 재배치하기 위한 심볼 위치 정보를 결정하는 제어기와,

상기 (n, k) 블록 부호를 구성하는 부호화 심볼들 각각을 상기 제어기가 결정한 심볼 위치 정보에 상응하게 IFHT기의 입력으로 재배치하는 심볼 배치기와,

상기 심볼 배치기에서 재배치한 심볼들을 입력하여 IFHT를 수행하는 IFHT기와,

상기 IFHT를 수행한 결과들중에서 최대 상관값을 가지는 (n, k) 블록 부호의 정보 비트를 복호 신호로 출력하는 비교 선택기를 포함함을 특징으로 하는 상기 복호 장치.

【청구항 2】

제1항에 있어서,

상기 생성 행렬 정보는 상기 (n, k) 블록 부호 생성을 위한 $k \times n$ 행렬을 나타내며, 상기 IFHT 크기는 상기 $k \times n$ 행렬에서 상위 m 행의 기저들을 IFHT기의 입력으로 사용하도록 제어하는 정보임을 특징으로 하는 상기 복호 장치.

【청구항 3】

제2항에 있어서,

상기 제어기는 상기 $k \times n$ 행렬의 제1열부터 순차적으로 제n열까지 각각 상위 m 엘리먼트를만을 고려하여 제1행의 엘리먼트를 최하위 비트로, 제m행의 엘리먼트를 최상위 비트로하여 이진 시퀀스들을 생성하고, 상기 생성한 이진 시퀀스들 각각의 십진수값을 생성하고, 상기 (n, k) 블록 부호의 부호화 심볼들중 첫 번째 부호화 심볼부터 순차적으로 제n 부호화 심볼까지 각각이 상기 제1열의 십진수값에 해당하는 상기 IFHT기의 입력으로 대응되고 순차적으로 제n열의 십진수값에 해당하는 상기 IFHT기의 입력으로 대응되도록 상기 심볼 위치 정보를 결정함을 특징으로 하는 상기 복호 장치.

【청구항 4】

제2항에 있어서,

상기 복호 장치는 상기 (n, k) 블록 부호를 소정 제어에 따라 제공되는 마스크들과 곱셈하여 상기 심볼 배치기로 출력하는 마스크 곱셈기를 더 포함함을 특징으로 하는 상기 복호 장치.

【청구항 5】

제4항에 있어서,

상기 제어기는 $k \times n$ 행렬에서 상위 m행의 기저들을 제외한 하위 $k-m$ 행의 기저들을 마스크 기저들로 사용하며, 상기 마스크 기저들을 상기 (n, k) 블록 부호에 적용된 변조 방식에 상

응하게 변조하여 생성된 마스크들을 상기 마스크 콥셈기로 제공함을 특징으로 하는 상기 복호 장치.

【청구항 6】

제3항에 있어서,

상기 심볼 배치기는;

상기 (n, k) 블록 부호를 입력되면 상기 제어기에서 제공되는 심볼 위치 정보에 상응하게 상기 (n, k) 블록 부호의 제1부호화 심볼부터 제 n 부호화 심볼까지 각각 해당하는 n 개의 가산기들 각각으로 스위칭하는 스위치와,

상기 IFHT기의 제1입력 내지 제 n 입력까지의 n 개의 입력들 각각에 연결되는 n 개의 메모리들과,

일단이 상기 스위치와 연결되며, 나머지 일단이 상기 n 개의 메모리들 각각과 연결되는 n 개의 가산기들을 포함함을 특징으로 하는 상기 복호 장치.

【청구항 7】

제6항에 있어서,

상기 심볼 배치기는 일단이 상기 n 개의 메모리들 각각에 연결되며, 나머지 일단이 상기 IFHT기에 연결되는 n 개의 스위치들을 포함하며, 상기 (n, k) 블록 부호의 부호어들 각각에 대한 심볼 재배치가 완료되면 상기 n 개의 스위치들 중 상기 IFHT기의 제1입력에 연결되는 스위치부터 순차적으로 상기 제 n 입력에 연결되는 스위치까지 상기 IFHT기로 순차적으로 연결하도록

제어함을 특징으로 하는 상기 복호 장치.

【청구항 8】

제6항에 있어서,

상기 심볼 배치기는 일단이 상기 n개의 메모리들 각각에 연결되며, 나머지 일단이 상기 IFHT기에 연결되는 병렬/직렬 변환기를 포함하며, 상기 병렬/직렬 변환기가 상기 (n, k) 블록 부호의 부호어를 각각에 대한 심볼 재배치가 완료되면 상기 n개의 메모리들 각각에 저장되어 있는 신호들을 상기 n개의 메모리들중 상기 IFHT기의 제1입력에 연결되는 메모리부터 순차적으로 상기 제n입력에 연결되는 메모리까지 직렬 변환하여 상기 IFHT기로 출력하도록 제어함을 특징으로 하는 상기 복호 장치.

【청구항 9】

k개의 정보 비트들과 n개의 출력 비트들을 가지는 (n, k) 블록 부호를 복호하는 장치에 있어서,

상기 (n, k) 블록 부호의 생성 행렬 정보를 가지고 상기 (n, k) 블록 부호를 역 고속 하다마드 변환(IFHT: Inverse Fast Hadamard Transform)하기 위한 IFHT 크기 정보와, 상기 (n, k) 블록 부호를 구성하는 부호화 심볼들 각각을 재배치하기 위한 심볼 위치 정보를 결정하는 제어기와,

상기 (n, k) 블록 부호를 구성하는 부호화 심볼들 각각을 상기 제어기가 결정한 심볼 위치 정보에 상응하게 IFHT기의 입력으로 재배치하는 심볼 배치기와,

상기 심볼 배치기에서 재배치한 심볼들을 입력하여 IFHT를 수행하는 IFHT기와,
상기 IFHT를 수행한 결과들중에서 최대 상관값을 가지는 (n, k) 블록 부호의 정보 비트
를 복호 신호로 출력하는 비교 선택기를 포함함을 특징으로 하는 상기 복호 장치.

【청구항 10】

제9항에 있어서,

상기 생성 행렬 정보는 상기 (n, k) 블록 부호 생성을 위한 $k \times n$ 행렬을 나타냄을 특징
으로 하는 상기 복호 장치.

【청구항 11】

제9항에 있어서,

상기 제어기는 상기 생성 행렬 정보를 가지고 IFHT를 수행할 경우의 연산량과, 시스템의
복잡도와, IFHT 수행 시간을 고려하여 상기 IFHT의 크기 정보를 결정하며, 상기 IFHT 크기 정
보는 상기 $k \times n$ 행렬에서 상위 m 행의 기저들을 IFHT기의 입력으로 사용하도록 제어하는 정보임
을 특징으로 하는 상기 복호 장치.

【청구항 12】

제11항에 있어서,

상기 제어기는 상기 $k \times n$ 행렬의 제1열부터 순차적으로 제 n 열까지 각각 상위 m 엘리먼트
들만을 고려하여 제1행의 엘리먼트를 최하위 비트로, 제 m 행의 엘리먼트를 최상위 비트로하여

이진 시퀀스들을 생성하고, 상기 생성한 이진 시퀀스들 각각의 십진수값을 생성하고, 상기 (n, k) 블록 부호의 부호화 심볼들중 첫 번째 부호화 심볼부터 순차적으로 제n 부호화 심볼까지 각각이 상기 제1열의 십진수값에 해당하는 상기 IFHT기의 입력으로 대응되고 순차적으로 제n열의 십진수값에 해당하는 상기 IFHT기의 입력으로 대응되도록 상기 심볼 위치 정보를 결정함을 특징으로 하는 상기 복호 장치.

【청구항 13】

제11항에 있어서,

상기 복호 장치는 상기 (n, k) 블록 부호를 소정 제어에 따라 제공되는 마스크들과 곱셈하여 상기 심볼 배치기로 출력하는 마스크 곱셈기를 더 포함함을 특징으로 하는 상기 복호 장치.

【청구항 14】

제12항에 있어서,

상기 제어기는 $k \times n$ 행렬에서 상위 m행의 기저들을 제외한 하위 $k-m$ 행의 기저들을 마스크 기저들로 사용하며, 상기 마스크 기저들을 상기 (n, k) 블록 부호에 적용된 변조 방식에 상응하게 변조하여 생성된 마스크들을 상기 마스크 곱셈기로 제공함을 특징으로 하는 상기 복호 장치.

【청구항 15】

제12항에 있어서,

상기 심볼 배치기는;

상기 (n, k) 블록 부호를 입력되면 상기 제어기에서 제공되는 심볼 위치 정보에 상응하게 상기 (n, k) 블록 부호의 제1부호화 심볼부터 제 n 부호화 심볼까지 각각 해당하는 n 개의 가산기들 각각으로 스위칭하는 스위치와,

상기 IFHT기의 제1입력 내지 제 n 입력까지의 n 개의 입력들 각각에 연결되는 n 개의 메모리들과,

일단이 상기 스위치와 연결되며, 나머지 일단이 상기 n 개의 메모리들 각각과 연결되는 n 개의 가산기들을 포함함을 특징으로 하는 상기 복호 장치.

【청구항 16】

제15항에 있어서,

상기 심볼 배치기는 일단이 상기 n 개의 메모리들 각각에 연결되며, 나머지 일단이 상기 IFHT기에 연결되는 n 개의 스위치들을 포함하며, 상기 (n, k) 블록 부호의 부호어들 각각에 대한 심볼 재배치가 완료되면 상기 n 개의 스위치들 중 상기 IFHT기의 제1입력에 연결되는 스위치부터 순차적으로 상기 제 n 입력에 연결되는 스위치까지 상기 IFHT기로 순차적으로 연결하도록 제어함을 특징으로 하는 상기 복호 장치.

【청구항 17】

제15항에 있어서,

상기 심볼 배치기는 일단이 상기 n개의 메모리들 각각에 연결되며, 나머지 일단이 상기 IFHT기에 연결되는 병렬/직렬 변환기를 포함하며, 상기 병렬/직렬 변환기가 상기 (n, k) 블록 부호의 부호어들 각각에 대한 심볼 재배치가 완료되면 상기 n개의 메모리들 각각에 저장되어 있는 신호들을 상기 n개의 메모리들중 상기 IFHT기의 제1입력에 연결되는 메모리부터 순차적으로 상기 제n입력에 연결되는 메모리까지 직렬 변환하여 상기 IFHT기로 출력하도록 제어함을 특징으로 하는 상기 복호 장치.

【청구항 18】

k개의 정보 비트들과 n개의 출력 비트들을 가지는 (n, k) 블록 부호를 복호하는 방법에 있어서,

상기 (n, k) 블록 부호의 생성 행렬 정보와, 상기 (n, k) 블록 부호를 역 고속 하다마드 변환(IFHT: Inverse Fast Hadamard Transform)하기 위한 IFHT 크기 정보를 가지고 상기 (n, k) 블록 부호를 구성하는 부호화 심볼들 각각을 재배치하기 위한 심볼 위치 정보를 결정하는 과정과,

상기 (n, k) 블록 부호를 구성하는 부호화 심볼들 각각을 상기 결정된 심볼 위치 정보에 상응하게 IFHT의 입력으로 재배치하는 과정과,

상기 재배치한 심볼들을 입력하여 IFHT를 수행하는 과정과,

상기 IFHT를 수행한 결과들중에서 최대 상관값을 가지는 (n, k) 블록 부호의 정보 비트를 복호 신호로 출력하는 과정을 포함함을 특징으로 하는 상기 복호 방법.

【청구항 19】

제18항에 있어서,

상기 생성 행렬 정보는 상기 (n, k) 블록 부호 생성을 위한 $k \times n$ 행렬을 나타내며, 상기 IFHT 크기 정보는 상기 $k \times n$ 행렬에서 상위 m 행의 기저들을 IFHT의 입력으로 사용하도록 제어하는 정보임을 특징으로 하는 상기 복호 방법.

【청구항 20】

제19항에 있어서,

상기 심볼 위치 정보를 결정하는 과정은 상기 $k \times n$ 행렬의 제1열부터 순차적으로 제 n 열 까지 각각 상위 m 엘리먼트들만을 고려하여 제1행의 엘리먼트를 최하위 비트로, 제 m 행의 엘리먼트를 최상위 비트로하여 이진 시퀀스들을 생성하고, 상기 생성한 이진 시퀀스들 각각의 십진 수값을 생성하고, 상기 (n, k) 블록 부호의 부호화 심볼들중 첫 번째 부호화 심볼부터 순차적으로 제 n 부호화 심볼까지 각각이 상기 제1열의 십진수값에 해당하는 상기 IFHT의 입력으로 대응되고 순차적으로 제 n 열의 십진수값에 해당하는 상기 IFHT의 입력으로 대응되도록 상기 심볼 위치 정보를 결정하는 것임을 특징으로 하는 상기 복호 방법.

【청구항 21】

제19항에 있어서,

상기 (n, k) 블록 부호를 소정 제어에 따라 제공되는 마스크들과 곱셈하여 상기 심볼 재배치하는 과정을 더 포함함을 특징으로 하는 상기 복호 방법.

【청구항 22】

제21항에 있어서,

상기 마스크들은 상기 $k \times n$ 행렬에서 상위 m 행의 기저들을 제외한 하위 $k-m$ 행의 기저들을 상기 (n, k) 블록 부호에 적용된 변조 방식에 상응하게 변조되어 생성된 것임을 특징으로 하는 상기 복호 방법.

【청구항 23】

k 개의 정보 비트들과 n 개의 출력 비트들을 가지는 (n, k) 블록 부호를 복호하는 방법에 있어서,

상기 (n, k) 블록 부호의 생성 행렬 정보를 가지고 상기 (n, k) 블록 부호를 역 고속 하다마드 변환(IFHT: Inverse Fast Hadamard Transform)하기 위한 IFHT 크기 정보와, 상기 (n, k) 블록 부호를 구성하는 부호화 심볼들 각각을 재배치하기 위한 심볼 위치 정보를 결정하는 과정과,

상기 (n, k) 블록 부호를 구성하는 부호화 심볼들 각각을 상기 결정된 심볼 위치 정보에 상응하게 IFHT의 입력으로 재배치하는 과정과,

상기 재배치한 심볼들을 입력하여 IFHT를 수행하는 과정과,
상기 IFHT를 수행한 결과들중에서 최대 상관값을 가지는 (n, k) 블록 부호의 정보 비트
를 복호 신호로 출력하는 과정을 포함함을 특징으로 하는 상기 복호 방법.

【청구항 24】

제23항에 있어서,

상기 생성 행렬 정보는 상기 (n, k) 블록 부호 생성을 위한 $k \times n$ 행렬을 나타냄을 특징
으로 하는 상기 복호 방법.

【청구항 25】

제23항에 있어서,

상기 IFHT의 크기 정보는 상기 생성 행렬 정보를 가지고 IFHT를 수행할 경우의
연산량과, 시스템의 복잡도와, IFHT 수행 시간을 고려하여 상기 IFHT의 크기 정보를 결정되며,
상기 IFHT 크기 정보는 상기 $k \times n$ 행렬에서 상위 m 행의 기저들을 IFHT의 입력으로 사용하도록
제어하는 정보임을 특징으로 하는 상기 복호 방법.

【청구항 26】

제25항에 있어서,

상기 심볼 위치 정보를 결정하는 과정은 상기 $k \times n$ 행렬의 제1열부터 순차적으로 제n열
까지 각각 상위 m 엘리먼트들만을 고려하여 제1행의 엘리먼트를 최하위 비트로, 제 m 행의 엘리

먼트를 최상위 비트로하여 이진 시퀀스들을 생성하고, 상기 생성한 이진 시퀀스들 각각의 십진 수값을 생성하고, 상기 (n, k) 블록 부호의 부호화 심볼들중 첫 번째 부호화 심볼부터 순차적으로 제 n 부호화 심볼까지 각각이. 상기 제1열의 십진수값에 해당하는 상기 IFHT의 입력으로 대응되고 순차적으로 제 n 열의 십진수값에 해당하는 상기 IFHT의 입력으로 대응되도록 상기 심볼 위치 정보를 결정하는 것임을 특징으로 하는 상기 복호 방법.

【청구항 27】

제25항에 있어서,

상기 (n, k) 블록 부호를 소정 제어에 따라 제공되는 마스크들과 곱셈하여 상기 심볼 재배치하는 과정을 더 포함함을 특징으로 하는 상기 복호 방법.

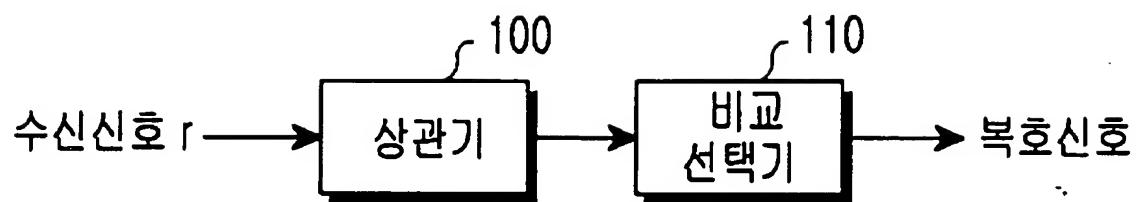
【청구항 28】

제27항에 있어서,

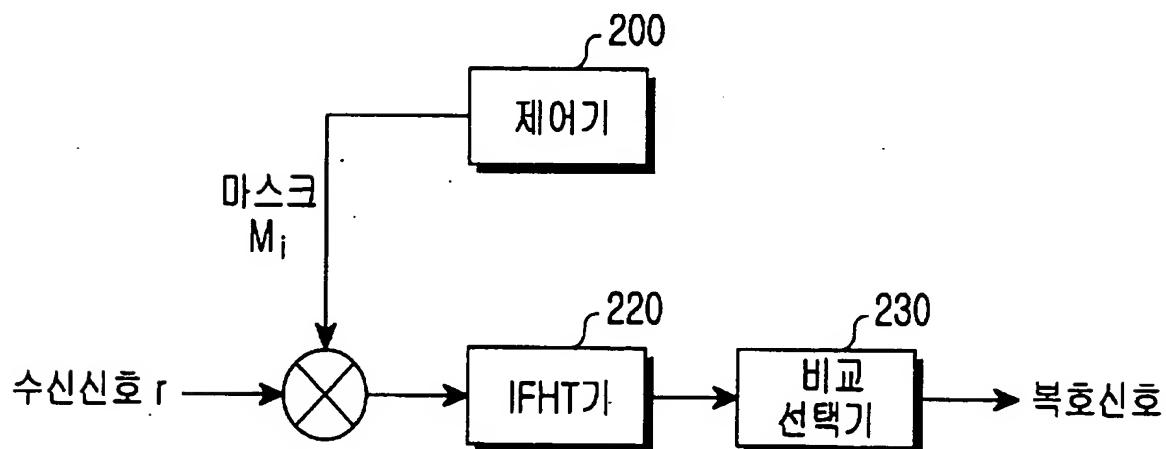
상기 마스크들은 상기 $k \times n$ 행렬에서 상위 m 행의 기저들을 제외한 하위 $k-m$ 행의 기저들을 상기 (n, k) 블록 부호에 적용된 변조 방식에 상응하게 변조되어 생성된 것임을 특징으로 하는 상기 복호 방법.

【도면】

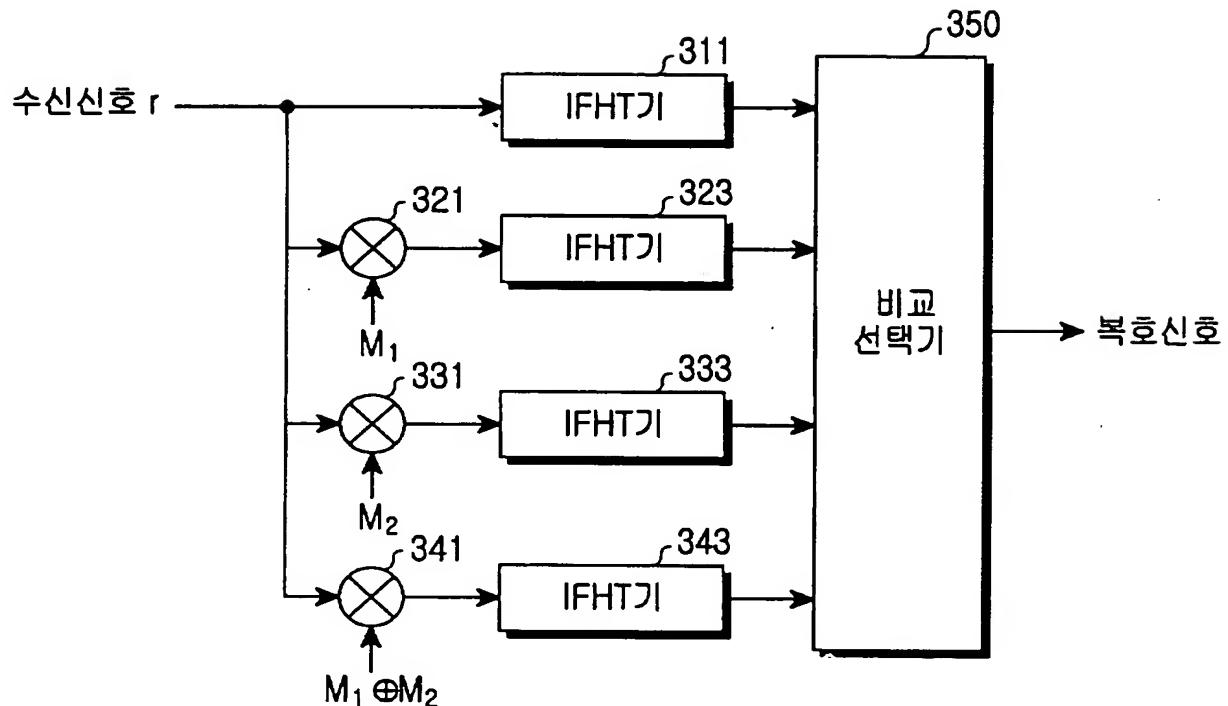
【도 1】



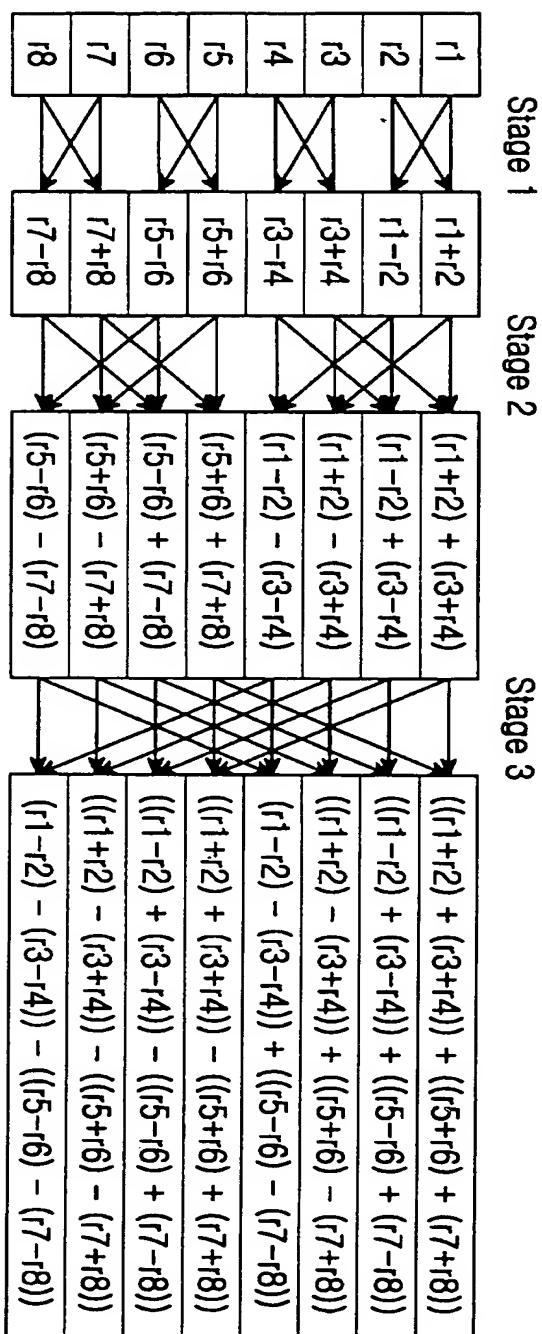
【도 2】



【도 3】



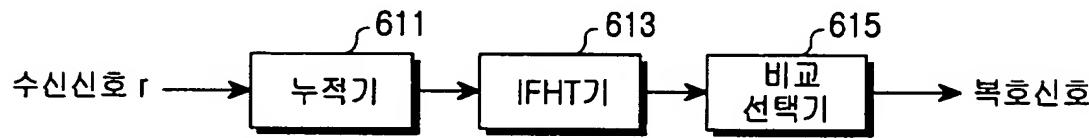
【도 4】



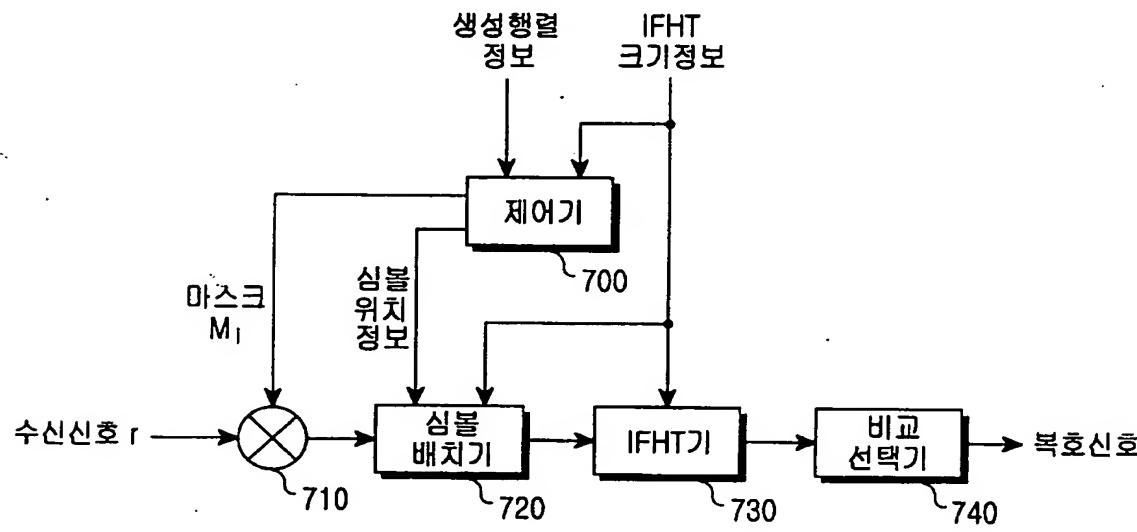
【도 5】



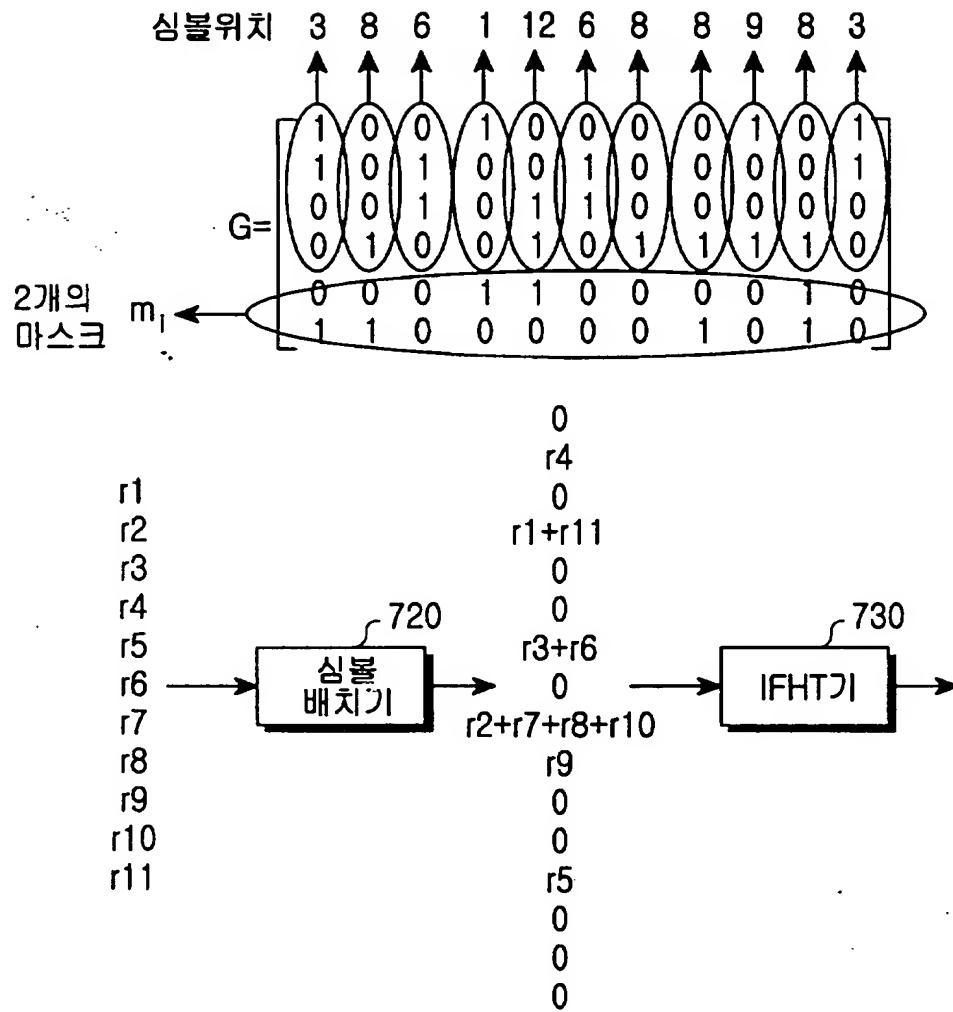
【도 6】



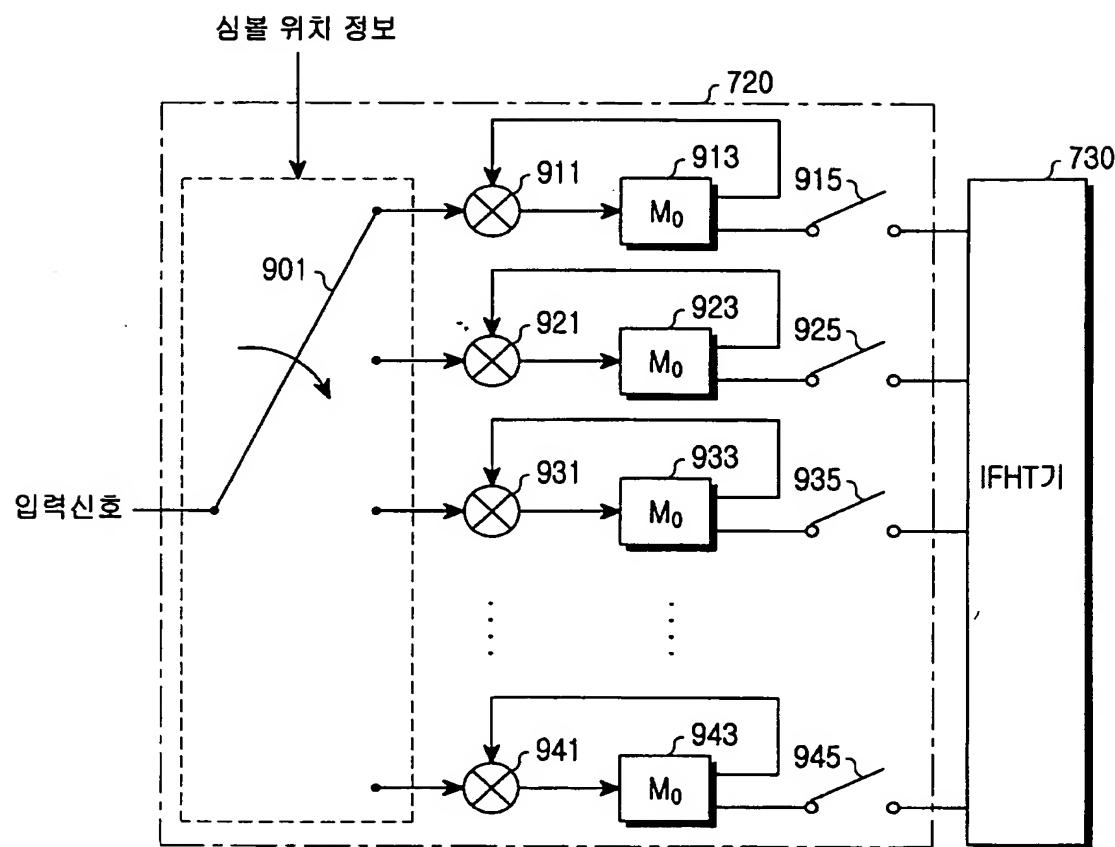
【도 7】



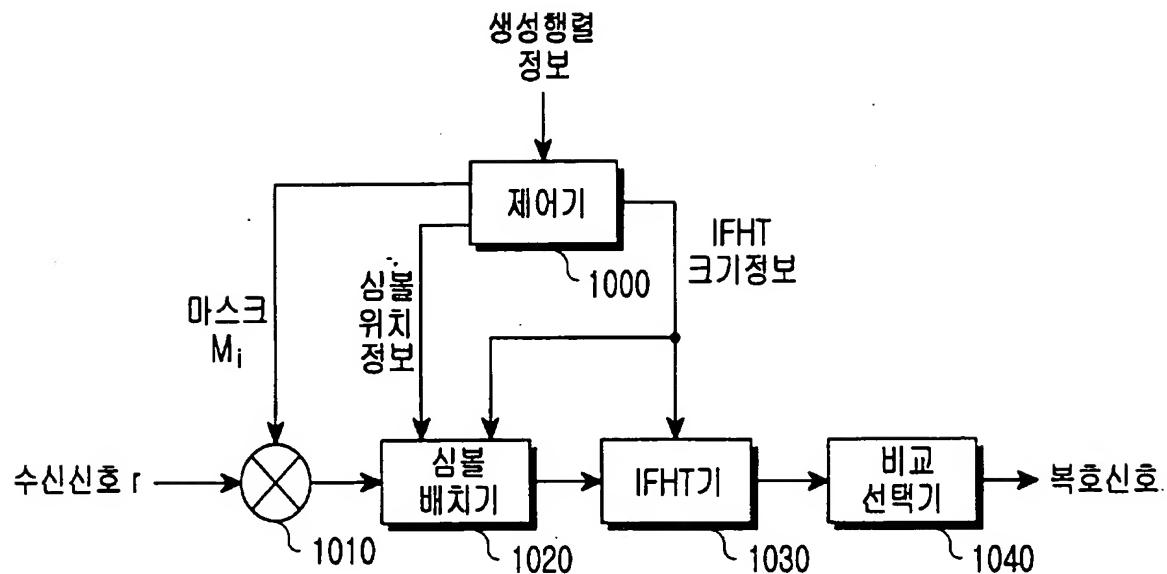
【도 8】



【도 9】



【도 10】



【도 11】

